TechnicalDocsChatController

September 5, 2026

TechnicalDocsChatController

Kind: Controller

Source: `atloria-monorepo/apps/api/src/technical-docs/technical-docs-chat.controller.ts`

Conversational grounded chat (Foundation F3): sessions with memory, SSE streaming,
and 👍/👎 feedback. Access mirrors /ask (public project open, else member token); the owning org pays the flat ask price per streamed question, and token usage is attributed via trackUsage(skipCreditDeduction).

`TechnicalDocsChatController` exposes grounded, conversational chat for technical documentation, including session memory, Server-Sent Events (SSE) streaming, and 👍/👎 response feedback. It applies the same access rules as `/ask`: publicly open projects can be queried without a token, while restricted projects require member authentication; streamed questions are charged to the owning organization and usage is tracked without an additional credit deduction.

Diagram

```
graph LR
  Client[Web client] -->|POST question| Controller[TechnicalDocsChatController]
  Controller --> Access[Project access check<br/>public or member token]
  Access --> Session[Chat session + memory]
  Session --> Grounding[Technical documentation grounding]
  Grounding --> Stream[SSE response stream]
  Stream --> Client
  Controller --> Billing[Charge owning org<br/>flat ask price]
  Controller --> Usage[trackUsage<br/>skipCreditDeduction]
  Client -->|👍 / 👎 feedback| Feedback[Feedback endpoint]
  Feedback --> Session
```

Usage

```ts
type ChatEvent = {
  type: "token" | "message" | "done" | "error";
  content?: string;
  sessionId?: string;
};

// Use the route configured by TechnicalDocsChatController.
async function askTechnicalDocs(
  projectId: string,
  question: string,
  sessionId?: string,
  accessToken?: string,
) {
  const response = await fetch("/technical-docs/chat", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Accept: "text/event-stream",
      ...(accessToken && {
        Authorization: `Bearer ${accessToken}`,
      }),
    },
    body: JSON.stringify({
      projectId,
      question,
      sessionId,
    }),
  });

  if (!response.ok || !response.body) {
    throw new Error(`Chat request failed: ${response.status}`);
  }

  const reader = response.body.getReader();
  const decoder = new TextDecoder();
  let answer = "";

  while (true) {
    const { done, value } = await reader.read();
    if (done) break;

    const chunk = decoder.decode(value, { stream: true });

    for (const line of chunk.split("\n")) {
      if (!line.startsWith("data: ")) continue;

      const event = JSON.parse(line.slice(6)) as ChatEvent;

      if (event.type === "token") {
        answer += event.content ?? "";
        process.stdout.write(event.content ?? "");
```

```
      }

      if (event.type === "done") {
        console.log("\nSession:", event.sessionId);
      }
    }
  }

  return answer;
}

await askTechnicalDocs(
  "project_123",
  "How do I configure authentication for this API?",
  undefined,
  process.env.ACCESS_TOKEN,
);
```

AI Coding Instructions

- Preserve `/ask`-equivalent authorization behavior: allow anonymous access only for public projects and require valid member access for restricted projects.
- Keep streamed responses SSE-compatible; send incremental events consistently and ensure errors are emitted in a format clients can handle.
- Reuse or extend existing session-memory services rather than storing conversation state directly in the controller.
- When recording usage for a streamed question, charge the owning organization once and call `trackUsage` with `skipCreditDeduction` enabled to avoid double charging.
- Validate feedback, project ownership, and session access before persisting 👍/👎 ratings.

Relationships

- MODULE_DECLARES → `createSession`
- MODULE_DECLARES → `stream`
- MODULE_DECLARES → `escalate`
- MODULE_DECLARES → `feedback`
- DEPENDS_ON → `TechDocsChatService`
- DEPENDS_ON → `TechnicalDocsService`
- DEPENDS_ON → `AIUsageService`

- DEPENDS_ON → `PrismaService`

Referenced By

- `TechnicalDocsModule` (MODULE_DECLARES)