FeedbackService

September 4, 2026

FeedbackService

Kind: Service

Source: `atloria-monorepo/apps/api/src/analytics/feedback.service.ts`

Per-doc helpfulness feedback. One row per (sessionId, documentSlug) — if a reader changes their mind and clicks the other thumb, the row upserts in place.

`FeedbackService` manages per-document helpfulness feedback for the analytics API. It stores one feedback record per `(sessionId, documentSlug)`, updates that record when a reader changes their vote, and exposes reporting methods for summaries, poorly rated documents, and recent negative comments.

Methods

Method	Signature	Returns	Description
`submit`	`submit(dto: SubmitFeedbackDto)`	`unknown`	Public — called by the widget beacon.
`getSummary`	`getSummary(projectId: string, days: unknown)`	`unknown`	Protected — overview stats for the project analytics dashboard.
`getWorstRated`	`getWorstRated(projectId: string, days: unknown, limit: unknown, minRatings: unknown)`	`unknown`	Protected — worst-rated docs, sorted ascending by

Method	Signature	Returns	Description
			helpfulness rate.
`getRecentNegativeComments`	`getRecentNegativeComments(projectId: string, days: unknown, limit: unknown)`	`unknown`	Protected — latest thumbs-down comments (the goldmine).

Dependencies

- `PrismaService`

When something fails

- `FeedbackService` handles failure in 1 place: it turns it into a return value in all 1.

Diagram

```
sequenceDiagram
    participant Client
    participant Controller as Feedback Controller
    participant Service as FeedbackService
    participant Database

    Client->>Controller: POST feedback (sessionId, documentSlug, helpful, comment)
    Controller->>Service: submit(feedbackDto)
    Service->>Database: Upsert by sessionId + documentSlug
    Database-->>Service: Saved feedback row
    Service-->>Controller: Feedback result
    Controller-->>Client: 200/201 response

    Client->>Controller: GET feedback summary
    Controller->>Service: getSummary()
    Service->>Database: Aggregate helpful/unhelpful votes
    Database-->>Service: Summary metrics
    Service-->>Controller: Summary response
```

Usage

```
import { Injectable } from '@nestjs/common';
import { FeedbackService } from './feedback.service';
```

```
@Injectable()
export class DocumentationAnalyticsService {
  constructor(private readonly feedbackService: FeedbackService) {}

  async recordFeedback() {
    await this.feedbackService.submit({
      sessionId: 'session_abc123',
      documentSlug: 'getting-started',
      helpful: false,
      comment: 'The installation steps are missing Docker instructions.',
    });

    const summary = await this.feedbackService.getSummary();
    const worstRated = await this.feedbackService.getWorstRated();
    const recentNegativeComments =
      await this.feedbackService.getRecentNegativeComments();

    return {
      summary,
      worstRated,
      recentNegativeComments,
    };
  }
}
```

AI Coding Instructions

- Preserve the `(sessionId, documentSlug)` uniqueness behavior when changing feedback persistence; submissions must upsert rather than create duplicate votes.
- Treat `submit()` as an update-capable operation: a reader may switch from helpful to unhelpful, or vice versa, within the same session.
- Keep aggregation logic in `getSummary()` and `getWorstRated()` database-backed where possible to avoid loading all feedback rows into memory.
- Ensure negative comments are handled defensively: comments may be optional, empty, or associated only with an unhelpful vote.
- Integrate this service through the analytics/controller layer rather than accessing feedback storage directly from documentation endpoints.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `AnalyticsController` (DEPENDS_ON)
- `AnalyticsModule` (MODULE_PROVIDES)
- `AnalyticsModule` (MODULE_EXPORTS)