FeatureDocsGeneratorService

September 4, 2026

FeatureDocsGeneratorService

Kind: Service

Source: `atloria-monorepo/apps/api/src/documentation/services/feature-docs-generator.service.ts`

Feature Documentation Generator Service

Generates comprehensive feature documentation using AI
Leverages Phase 2.1 & 3 context (state machines, UI interactions)

`FeatureDocsGeneratorService` generates comprehensive feature-level documentation by combining AI output with existing application context such as state machines and UI interaction data. It is part of the backend documentation pipeline and exposes `generateFeatureDocumentation()` to produce a structured `FeatureDocumentation` result for a feature.

Methods

Method	Signature	Returns	Description
`generateFeatureDocumentation`	`generateFeatureDocumentation(feature: DetectedFeature, options: {		

```
  generatorProvider?: 'azure-claude' | 'azure-openai';
  reviewerProvider?: 'azure-claude' | 'azure-openai';
  enableReviewer?: boolean;
  businessContext?: any;
})` | `Promise<FeatureDocumentation>` | Generate feature documentation with AI |
```

Dependencies

- `AzureClaudeProvider`
- `AzureOpenAIProvider`

Where it refuses work

- `FeatureDocsGeneratorService` stops the work with an early return when `feature.services.length === 0`.
- `FeatureDocsGeneratorService` stops the work with an early return when `feature.stateMachines.length === 0`.
- `FeatureDocsGeneratorService` stops the work with an early return when `feature.uiInteractions.length === 0`.

Diagram

```
sequenceDiagram
  participant Caller as Documentation Workflow
  participant Service as FeatureDocsGeneratorService
  participant Context as Phase 2.1 / 3 Context
  participant AI as AI Provider

  Caller->>Service: generateFeatureDocumentation(feature)
  Service->>Context: Load state machine and UI interaction context
  Context-->>Service: Feature context
  Service->>AI: Generate documentation prompt with context
  AI-->>Service: Generated feature documentation
  Service-->>Caller: Promise<FeatureDocumentation>
```

Usage

```
import { Injectable } from '@nestjs/common';
import { FeatureDocsGeneratorService } from './documentation/services/feature-docs-generator.se

@Injectable()
export class DocumentationJobService {
  constructor(
    private readonly featureDocsGenerator: FeatureDocsGeneratorService,
  ) {}

  async generateFeatureDocs() {
    const documentation =
      await this.featureDocsGenerator.generateFeatureDocumentation();

    return documentation;
  }
}
```

AI Coding Instructions

- Preserve the service's role as an orchestration layer: collect feature context, construct AI input, and return a `FeatureDocumentation` result.
- Include Phase 2.1 state-machine data and Phase 3 UI interaction context when generating prompts; avoid producing documentation from incomplete feature metadata alone.
- Keep AI prompt construction deterministic and structured so generated documentation remains consistent across runs.
- Handle AI provider failures, missing context, and malformed responses before returning documentation to callers.
- Register and inject the service through NestJS dependency injection rather than creating it manually.

Relationships

- DEPENDS_ON → `AzureClaudeProvider`
- DEPENDS_ON → `AzureOpenAIProvider`

Referenced By

- `DocumentationModule` (MODULE_PROVIDES)
- `DocumentationModule` (MODULE_EXPORTS)