CaptureController

September 4, 2026

CaptureController

Kind: Controller

Source: `atloria-monorepo/apps/api/src/capture/capture.controller.ts`

Member-gated capture endpoints. A project member records a walkthrough of THEIR product; we store the recipe (selectors + actions), never fabricated screenshots and never secret input values.

`CaptureController` exposes member-gated API endpoints for recording product walkthroughs within a project. It validates that the caller belongs to the target project, then persists a capture recipe made of selectors and user actions rather than fabricated screenshots or sensitive input values. It acts as the HTTP boundary between authenticated clients and the capture workflow services.

Diagram

```
graph LR
  Member[Authenticated project member] --> Controller[CaptureController]
  Controller --> Guard[Authentication and membership guard]
  Guard -->|Authorized| CaptureService[Capture service]
  Guard -->|Denied| Forbidden[403 response]
  CaptureService --> Sanitizer[Recipe sanitization]
  Sanitizer --> Recipe[(Stored walkthrough recipe)]
  Recipe --> Data[Selectors and actions only]
  Data -. never stores .-> Sensitive[Secret input values or fabricated screenshots]
```

Usage

```
// Example client request for recording a walkthrough recipe.
// The authenticated user must be a member of the referenced project.

const response = await fetch("/api/capture", {
  method: "POST",
  headers: {
```

```
    "Content-Type": "application/json",
    Authorization: `Bearer ${accessToken}`,
  },
  body: JSON.stringify({
    projectId: "project_123",
    actions: [
      {
        type: "click",
        selector: "[data-testid='start-tour']",
      },
      {
        type: "navigate",
        url: "/dashboard",
      },
      {
        type: "click",
        selector: "[data-testid='create-item']",
      },
    ],
  }),
});

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

const capture = await response.json();
console.log("Stored capture recipe:", capture);
```

AI Coding Instructions

- Keep all capture endpoints protected by the existing authentication and project-membership authorization flow.
- Store replayable selectors and action metadata only; never persist raw secret values, credentials, payment details, or fabricated screenshots.
- Validate capture payloads at the controller boundary using the project's DTO and validation conventions before passing data to services.
- Keep authorization and HTTP concerns in `CaptureController`; put recipe processing, sanitization, and persistence logic in the capture service layer.
- When adding new action types, ensure they are explicitly supported by the recipe schema and safely handled by downstream replay consumers.

Relationships

- MODULE_DECLARES → `create`

- MODULE_DECLARES → `list`
- MODULE_DECLARES → `get`
- MODULE_DECLARES → `updateSteps`
- MODULE_DECLARES → `reshootStep`
- MODULE_DECLARES → `reshootLocales`
- MODULE_DECLARES → `remove`
- MODULE_DECLARES → `draft`
- DEPENDS_ON → `CaptureService`

Referenced By

- `CaptureModule` (MODULE_DECLARES)