Home & Insights Interactivity Map

STUDENT CRM ANGULAR PROJECT

Generated 2025-11-01

Goal. Show how actions on the Home page update the shared student data and how the Insights page responds through navigation, routing, and data transformations.

Key Building Blocks

ELEMENT	FILE & RESPONSIBILITY
Navigation Shell	`src/app/app.html`: header links expose Home and Insights routes. Clicking Insights routes to `/students/stats` without reloading.
Home Page	`src/app/home/home.ts` and `home.html`: load students via `StudentService`, render the add form and list, react to child events to mutate the store.
Insights Page	`src/app/students/student-stats/student-stats.ts` and `student-stats.html`: read the service list, aggregate totals, respond to query parameters.
Shared State	`src/app/core/student.ts`: `StudentService` exposes `list()`, `add()`, `toggleActive()`; both pages rely on the same instance (provided in root).

Interactivity Flow

1. **Navigation.** The header link `<a routerLink="/students/stats" routerLinkActive="active">` switches the router outlet from Home to Insights.
2. **Home actions.**
   - Submitting the add form calls `AddStudentComponent.submit()`, which emits a `create` event containing the new student's fields.

- `HomeComponent.onCreate` receives the event, calls `StudentService.add()`, and replaces its `students` array with the returned list.
- Clicking "Toggle Active" on a student card emits the student id to `HomeComponent.onToggleActive`, which calls `StudentService.toggleActive()` and refreshes the list.

3. **Insights recompute.**
   - When Insights loads, `StudentStatsComponent.ngOnInit()` calls `StudentService.list()` to get the current snapshot.
   - The component reduces the array into totals per track and overall active/inactive counts by running `recomputeStats()`.
   - A subscription to `route.queryParamMap` keeps `selectedTrack` in sync with the URL and filters `students` accordingly.
4. **Feedback loop.**
   - Service mutations triggered on Home are immediately reflected in subsequent `list()` calls.
   - Navigating between Home and Insights demonstrates the updated aggregates without additional plumbing.

Event & Data Map

```
[AddStudentComponent] submit()
  → emits { name, track, active }
[HomeComponent.onCreate]
  → StudentService.add(data)
[StudentService]
  → updates store and returns fresh array
[HomeComponent]
  → sets this.students = svc.list()
[StudentStatsComponent]
  → on navigation, calls svc.list()
  → recomputeStats() builds track summaries
```

Query Parameter Interactivity

Insights exposes filter chips that call `selectTrack(track)`, updating the router's query string while staying on the same component:

```
this.router.navigate([], {
  relativeTo: this.route,
  queryParams: track ? { track } : { track: null },
  queryParamsHandling: 'merge'
});
```

Because `StudentStatsComponent` subscribes to `queryParamMap`, the UI reacts instantly to URL state, demonstrating how navigation can drive data presentation without extra service calls.

Teaching Takeaways

- Both screens share a single source of truth through dependency injection; no manual syncing is needed.
- Router navigation provides the context switch while preserving application state.
- Query parameters act as a bookmarking mechanism for the Insights filter.
- Outputs on Home (create/toggle events) are the triggers that keep analytics fresh.

Use the file references above when guiding students through the live code.