title: Student CRM â Home & Insights Walkthrough
paginate: true
size: A4

WELCOME!

This guide breaks down how the **Home** screen and the **Insights window** (stats dashboard) work together in the Student CRM project. We’ll use plain language so you can explain it easily to students who are new to Angular.

1. WHAT ARE THESE SCREENS?

â¢ **Home** is where teachers add students and flip their active/inactive status.
â¢ **Insights** is a dashboard that reads the same list of students and turns it into totals, charts, and filters.
â¢ Both pages live inside one Angular app, so switching between them does **not** reload the browser.

NOTE: Think of Home as the "data entry" room and Insights as the "summary" room that always peeks at the same notebook.

2. SHARED TOOLBOX

â¢ Router links â ‘src/app/app.component.html‘ â Links labelled **Home** and **Insights** move between pages without leaving the single-page app.
â¢ Student service â ‘src/app/core/student.service.ts‘ â Stores every student in memory and offers ‘list()‘, ‘add()‘, and ‘toggleActive()‘ helper methods.
â¢ Home component â ‘src/app/home/home.component.ts‘ + ‘.html‘ â Calls the service to show and change students.
â¢ Stats component â ‘src/app/students/student-stats/student-stats.component.ts‘ + ‘.html‘ â Reads from the service, groups the data, and shows the dashboard UI.

3. NAVIGATION FLOW

1. Inside the header, the link ‘<a routerLink="/students/stats">Insights</a>‘ tells Angular to swap the page content.
2. Angular looks in ‘AppRoutingModule‘ and sees that ‘/students/stats‘ should load ‘StudentStatsComponent‘.
3. The router outlet switches to the stats component instantly. No refresh, no data loss.
4. The **Back to Home** button is simply ‘routerLink="/"‘, so students can jump back with one click.

NOTE: Tip: Because routing happens in the browser, any typed form values on Home stay put until you leave the page on purpose.

4. HOME IN SLOW MOTION

1. ‘HomeComponent‘ runs ‘ngOnInit()‘ â calls ‘studentService.list()‘ â stores that array in ‘this.students‘.
2. When the **Add Student** form fires its ‘(create)‘ output, ‘onCreate()‘ sends the payload to ‘studentService.add()‘.
3. The service returns an updated array. Home replaces ‘this.students‘ so the new person shows up immediately.
4. Toggling a student button calls ‘studentService.toggleActive(id)‘, which flips the ‘active‘ flag and returns the fresh list again.

Students only need to remember: *"Home always asks the service to do the work, then grabs the updated list."*

5. INSIGHTS IN SLOW MOTION

1. ‘StudentStatsComponent‘ also calls ‘studentService.list()‘ when it loads.
2. It runs a helper named ‘recomputeStats()‘ that counts:
   - total students
   - active vs. inactive
   - how many students are in each track
3. It watches the router’s query string for ‘?track=science‘ and keeps that value in ‘selectedTrack‘.
4. The template shows filter chips. Clicking a chip calls ‘selectTrack(track)‘, which updates the query string. The getter ‘filteredStudents‘ then returns only the matching students.

NOTE: Because filters live in the URL, bookmarking the page keeps the same selection when you come back.

6. PUTTING IT TOGETHER

[Home] form submit â StudentService.add() â shared store updates

[Home] toggle active â StudentService.toggleActive() â shared store updates

Navigate to Insights â StudentService.list() â stats recompute from the same store

â¢ There is **no special messaging** between Home and Insights. The service is the single source of truth.
â¢ Every time you open Insights, it re-reads the student list, so any changes from Home appear instantly.

7. CLASSROOM TALKING POINTS

â¢ Emphasise the role of the service: *"Both components borrow the same clipboard."*
â¢ Show students how navigation is just ‘routerLink‘ attributesâ no extra JavaScript is required.
â¢ Highlight the benefit of query parameters: they are easy to share or bookmark.
â¢ Encourage learners to trace data in the TypeScript files first, then peek at the HTML to

see how the data is displayed.

8. QUICK RECAP

1. Home and Insights are two components routed inside one Angular app.
2. Both rely on ‘StudentService‘, which owns the student list.
3. Home changes the data; Insights reads and summarises it.
4. The router keeps navigation smooth and remembers filter state through query parameters.

You now have a beginner-friendly story that links the Home actions to the Insights dashboard. Happy teaching!