Student CRM Test Failures – Troubleshooting Notes

Generated 2025-11-01

Summary. Three Jasmine/Karma unit tests fail because the specs still assume certain components are standalone while the application now declares them in an NgModule. Adjust the TestBed setup (or revert the components back to standalone) to resolve the errors.

The Failing Specs

SPEC FILE	ERROR MESSAGE	ROOT CAUSE
`src/app/students/student-list/student-list.spec.ts`	"Unexpected `HostComponent` found in the `declarations` array … the component is marked as standalone and can't be declared in any NgModule."	The spec registers `StudentListComponent` and `StudentCardComponent` in the `imports` array, assuming they are standalone. In the project they are declared inside `StudentsModule`, so TestBed expects them in `declarations` (or the module should be imported wholesale).
`src/app/students/student-card/student-card.spec.ts`	"Unexpected directive `StudentCardComponent2` imported by the module `DynamicTestModule`.	Same mismatch: the spec imports the component as if it were standalone, but the app declares it in an NgModule. Karma

SPEC FILE	ERROR MESSAGE	ROOT CAUSE
	Please add an `@NgModule` annotation.”	interprets the class as a directive without module metadata and throws.
`student-list.spec.ts` (secondary issue)	Missing closing `});` for the first `it` block.	Not the current failure, but the suite will break once the module issue is fixed, so add the closing brace.

Why It Happened

The application moved its student components into `StudentsModule` (`src/app/students/students-module.ts`) with `standalone: false`. Unit tests, however, were written back when these components were standalone. In Angular 15+, standalone components go into the `imports` array; NgModule-declared components belong in `declarations`. Mixing the two signals yields the runtime errors you observed.

How to Fix It

1. **Update the TestBed setup to match module usage.**
   - Option A: move the components from `imports` to `declarations` and add `CommonModule` / `FormsModule` as needed.
   - Option B: import `StudentsModule` directly so TestBed reuses the production declarations.
2. **Or revert the components to standalone.** If you set `standalone: true` again and adjust the app module accordingly, the existing tests will work without changes. (This trades classroom goals for compatibility.)
3. **Close the open test block.** Add the missing `});` after the first `it` in `student-list.spec.ts` so the second test executes.

Suggested Classroom Talking Points

- Contrast TestBed configuration for standalone components vs. NgModule-declared ones.

- Show how shared modules (like `StudentsModule`) can simplify test setup.
- Discuss keeping specs in sync with refactors to avoid brittle tests.

Once you align the test configuration with the module structure, rerun `ng test` to confirm all specs pass.