Student CRM — Classroom Build Guide

Use this handout to lead a live build of the Student CRM application with your students. Every step works with Angular CLI 20 in non-standalone mode and results in the exact project now running locally.

> **Prerequisites:** Node 18+, Angular CLI 20+, and PowerShell (Windows). Run all shell commands from a terminal unless noted otherwise.

1. Create the Angular workspace

```
ng new student-crm --standalone=false --routing --style=css
cd student-crm
```

2. Wire the bootstrap file

Edit `src/main.ts` so Angular bootstraps `AppModule`.

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';

platformBrowserDynamic()
  .bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

3. Generate the Home component

```
ng g component home --standalone=false
```

4. Create core services

Generate services and replace their contents with the following code.

```
ng g service core/student
ng g service core/logger
```

```
// src/app/core/student.service.ts
import { Injectable } from '@angular/core';

export interface Student {
  id: number;
  name: string;
  track: 'Front-end' | 'Back-end' | 'Data' | 'DevOps';
  active: boolean;
}

@Injectable({ providedIn: 'root' })
export class StudentService {
  private store: Student[] = [
    { id: 1, name: 'Alice', track: 'Front-end', active: true },
    { id: 2, name: 'Bob', track: 'Data', active: false }
  ];

  list(): Student[] { return [...this.store]; }

  add(s: Omit<Student, 'id'>): Student[] {
    const id = Math.max(0, ...this.store.map(x => x.id)) + 1;
    this.store = [...this.store, { id, ...s }];
    return this.list();
  }

  toggleActive(id: number): Student[] {
    this.store = this.store.map(s =>
      s.id === id ? { ...s, active: !s.active } : s
    );
    return this.list();
  }
}
```

```
// src/app/core/logger.service.ts
import { Injectable } from '@angular/core';

@Injectable({ providedIn: 'root' })
export class LoggerService {
  log(message: string) {
    console.log(`[LOG] ${message}`);
  }
}
```

5. Generate the Students feature module and components

```
ng g module students
ng g component students/add-student --standalone=false
ng g component students/student-card --standalone=false
ng g component students/student-list --standalone=false
```

6. Wire the Students module

Replace `src/app/students/students.module.ts` with:

```
// src/app/students/students-module.ts
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

import { AddStudentComponent } from './add-student/add-student';
import { StudentCardComponent } from './student-card/student-card';
import { StudentListComponent } from './student-list/student-list';

@NgModule({
  declarations: [
    AddStudentComponent,
    StudentCardComponent,
    StudentListComponent
  ],
  imports: [CommonModule, FormsModule],
  exports: [AddStudentComponent, StudentListComponent]
})
export class StudentsModule {}
```

7. Implement the Students components

```
// src/app/students/add-student/add-student.ts
import { Component, EventEmitter, Output } from '@angular/core';

@Component({
  selector: 'app-add-student',
  templateUrl: './add-student.html',
  styleUrls: ['./add-student.css']
})
export class AddStudentComponent {
  @Output() create = new EventEmitter<{ name: string; track: string; active: boolean }>();

  name = '';
  track: 'Front-end' | 'Back-end' | 'Data' | 'DevOps' = 'Front-end';
  active = true;
```

```
  submit() {
    if (!this.name.trim()) return;
    this.create.emit({ name: this.name, track: this.track, active: this.active });
    this.name = '';
    this.active = true;
  }
}
```

```
<!-- src/app/students/add-student/add-student.html -->
<form class="add-form" (ngSubmit)="submit()">
  <input class="txt" name="name" [(ngModel)]="name" placeholder="Student name" required>
  <select class="sel" name="track" [(ngModel)]="track">
    <option>Front-end</option>
    <option>Back-end</option>
    <option>Data</option>
    <option>DevOps</option>
  </select>
  <label class="chk">
    <input type="checkbox" name="active" [(ngModel)]="active"> Active
  </label>
  <button class="btn" type="submit" [disabled]="!name">Add</button>
</form>
```

```
/* src/app/students/add-student/add-student.css */
.add-form { display: flex; gap: 8px; align-items: center; flex-wrap: wrap; }
.txt, .sel { padding: 8px; border-radius: 6px; border: 1px solid #ccc; }
.chk { display: flex; align-items: center; gap: 4px; }
.btn { padding: 8px 12px; border-radius: 6px; background: #18243d; color: #fff; border: non
```

```
// src/app/students/student-card/student-card.ts
import { Component, EventEmitter, Input, OnDestroy, OnInit, Output } from '@angular/core';
import { Student } from '../../core/student.service';

@Component({
  selector: 'app-student-card',
  templateUrl: './student-card.html',
  styleUrls: ['./student-card.css']
})
export class StudentCardComponent implements OnInit, OnDestroy {
  @Input({ required: true }) student!: Student;
  @Output() toggle = new EventEmitter<number>();

  ngOnInit() { console.log(`Card init: ${this.student?.name}`); }
```

```
  ngOnDestroy() { console.log(`Card destroy: ${this.student?.name}`); }
}
```

```
<!-- src/app/students/student-card/student-card.html -->
<div class="card">
  <div class="row1">
    <h5 class="title">{{ student.name }}</h5>
    <span class="badge" [class.on]="student.active" [class.off]="!student.active">
      {{ student.active ? 'ACTIVE' : 'INACTIVE' }}
    </span>
  </div>
  <p class="track">Track: <b>{{ student.track }}</b></p>
  <div class="actions">
    <button class="btn-outline" (click)="toggle.emit(student.id)">Toggle Active</button>
  </div>
</div>
```

```
/* src/app/students/student-card/student-card.css */
.card { border: 1px solid #e5e5e5; border-radius: 8px; padding: 10px; box-shadow: 0 1px 2px
.row1 { display: flex; justify-content: space-between; align-items: center; }
.title { margin: 0; }
.badge { padding: 2px 8px; border-radius: 999px; font-size: 12px; }
.badge.on { background: #e7f7ee; color: #0b6b3a; }
.badge.off { background: #f1f1f1; color: #666; }
.track { margin: 6px 0 0; }
.actions { margin-top: 8px; }
.btn-outline { padding: 6px 10px; border: 1px solid #ccc; border-radius: 6px; background: #
```

```
// src/app/students/student-list/student-list.ts
import { Component, EventEmitter, Input, Output } from '@angular/core';
import { Student } from '../../core/student.service';

@Component({
  selector: 'app-student-list',
  templateUrl: './student-list.html',
  styleUrls: ['./student-list.css']
})
export class StudentListComponent {
  @Input() students: Student[] = [];
  @Output() toggleActive = new EventEmitter<number>();
}
```

```
<!-- src/app/students/student-list/student-list.html -->
<p *ngIf="!students.length" class="muted">No students yet.</p>
```

```
<div class="grid" *ngIf="students.length">
  <div class="col" *ngFor="let s of students">
    <app-student-card
      [student]="s"
      (toggle)="toggleActive.emit($event)">
    </app-student-card>
  </div>
</div>
```

```
/* src/app/students/student-list/student-list.css */
.muted { color: #666; }
.grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(260px, 1fr)); gap: 1
.col { display: block; }
```

8. Finish the Home feature

```
// src/app/home/home.ts
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Student, StudentService } from '../core/student.service';
import { LoggerService } from '../core/logger.service';

@Component({
  selector: 'app-home',
  templateUrl: './home.html',
  styleUrls: ['./home.css']
})
export class HomeComponent implements OnInit, OnDestroy {
  students: Student[] = [];

  constructor(private svc: StudentService, private log: LoggerService) {}

  ngOnInit(): void {
    this.students = this.svc.list();
    this.log.log('Home initialized');
  }

  addStudent(data: Omit<Student, 'id'>) {
    this.students = this.svc.add(data);
  }

  toggleStatus(id: number) {
    this.students = this.svc.toggleActive(id);
  }

  ngOnDestroy(): void {
    this.log.log('Home destroyed');
```

```
  }
}
```

```
<!-- src/app/home/home.html -->
<div class="header">
  <h2 class="m-0">Students</h2>
  <span class="badge">{{ students.length }}</span>
</div>

<app-add-student (create)="addStudent($event)"></app-add-student>

<app-student-list
  [students]="students"
  (toggleActive)="toggleStatus($event)">
</app-student-list>
```

```
/* src/app/home/home.css */
.header { display: flex; gap: 8px; align-items: center; margin-bottom: 8px; }
.badge { background: #eee; padding: 2px 8px; border-radius: 999px; }
```

9. Configure routing

Update `src/app/app-routing.module.ts` so the home page renders by default.

```
// src/app/app-routing-module.ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: '**', redirectTo: '' }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

10. Replace the CLI shell

Swap out the Angular CLI placeholder with the layout used in the finished app. Update both template and styles.

```
// src/app/app.html
<main class="app">
  <header class="app__header">
    <h1 class="app__title">Student CRM</h1>
    <nav class="app__nav">
      <a routerLink="/" routerLinkActive="active" [routerLinkActiveOptions]="{ exact: true
        Home
      </a>
    </nav>
  </header>
  <section class="app__content container">
    <router-outlet></router-outlet>
  </section>
</main>
```

```
// src/app/app.css
.app {
  min-height: 100vh;
  background: #f5f6f8;
  display: flex;
  flex-direction: column;
}

.app__header {
  background: #18243d;
  color: #fff;
  padding: 16px 24px;
  display: flex;
  align-items: center;
  justify-content: space-between;
  gap: 16px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.12);
}

.app__title {
  margin: 0;
  font-size: 20px;
  font-weight: 600;
}

.app__nav a {
  color: #b8c1d1;
  text-decoration: none;
  font-weight: 500;
  padding: 4px 8px;
  border-radius: 4px;
}
```

```
.app__nav a.active,
.app__nav a:hover {
  color: #fff;
  background: rgba(255, 255, 255, 0.16);
}

.app__content {
  flex: 1;
  padding: 32px 24px;
}
```

11. Finalize AppComponent logic

Edit `src/app/app.ts` to hold the app title.

```
// src/app/app.ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.html',
  styleUrls: ['./app.css']
})
export class App {
  title = 'Student CRM';
}
```

```
// src/app/app-module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing-module';
import { App } from './app';
import { HomeComponent } from './home/home';
import { StudentsModule } from './students/students-module';

@NgModule({
  declarations: [App, HomeComponent],
  imports: [BrowserModule, AppRoutingModule, StudentsModule],
  providers: [],
  bootstrap: [App]
})
export class AppModule {}
```

12. Run the application

```
npm install
ng serve -o
```

13. Demo talking points

- Fresh page load shows two seeded students, proving the service wiring.
- The badge count reacts to adds and toggles thanks to immutable updates.
- `LoggerService` messages appear in the console, illustrating cross-cutting services.
- Each card logs lifecycle hooks (`ngOnInit`/`ngOnDestroy`) when the list changes.

With these files in place the build passes (`npm run build`) and the UI matches the finished Student CRM dashboard.