Expeditors – Angular Academy
Capstone 04 Instructions & Requirements

Create an Angular UI that supports the CRUD operations exposed by the Java API that you built out in the previous 3 capstones. As a reminder, the API provides a Spring Boot-based set of REST endpoints for maintaining a Music and Entertainment management system (tracks and artists). In this capstone, you will primarily be focusing on list views and "detail" views for the key business entities in your application (including forms for creating new and editing existing).

- Students will work in teams designing and building the application
    - Design work should be done as a team – each member of the team should be aligned on any design decision made
    - Implementation individually or as a group using paired or mob programming; if executed as a group, it is critical that every member of the team get an opportunity for "hands on keyboard" coding
    - Each team should have a short meeting at a fixed time every day to talk about status and any issues or blockers that team members are facing

Features:

- Provide list/grid views for each major domain object defined in your application (e.g., for artists, for tracks, etc.)
- Provide forms for each major domain object defined in your application which will allow the user to view details for an existing entry, edit an existing entry, or create a new entry
- You can decide how sophisticated you want your displays to be – e.g., do you want to include options that allow the user to associate specific tracks from the set of available tracks with an artist that you are interacting with and/or vice versa?
- Recommendation: execute as a set of "mini-sprints"
    - Complete one view/feature from end-to-end before starting on the next (e.g., get the list view for artists working before moving on to another feature)
    - You can decide if you want styling to be a part of the feature delivered or you can keep styling as a separate feature applied after CRUD operations on all pages are functioning correctly
    - Due to time constraints and the number of features to be implemented in the time available, authentication/authorization (through the UI) should likely be left as the last feature implemented (if time permits)
    - This means that you will need to remove any authentication (temporarily) from your API while you're building the initial features

Tech Stack:

- Angular
- Java
- Spring/Spring Boot
- PostgreSQL database – containerized or running locally
- JPA/Hibernate

- JWTs – using “roll your own” JWT generation/validation components or via KeyCloak (OPTIONAL)

Deliverables:

- A functioning UI built using Angular that supports CRUD operations against the business entities defined by your REST API
- OPTIONAL: Styling
- OPTIONAL: User registration/login
- A presentation on the completed application, including a summary of what you learned throughout the program and where you hope to go next in your learning and implementation journey