Homework Assignment #4 and #5

READ EVERYTHING

Assignment Summary

Homework 4 and 5 are combined assignments to be turned in as one submission.

You will be building additional features on the project from homework 3. Changes to be incorporated include:

- Introducing routing
- Adding ability to add, edit and delete movies
- Abstracting rating into its own component
- Minor enhancements to list

The full list of requirements is itemized on the next page.

Watch video of completed assignment here: http://nimb.ws/dpvYuT

Instead of using your own Homework #3, start from my completed solution which I have modified slightly to help you with this assignment. You can download it here: http://tinyurl.com/z3ys9n9.

Be sure to look over the exercises from class as reference. As always, email me if you are stuck, but don't wait till the last minute. I am also available before or after class with notice.

Submission

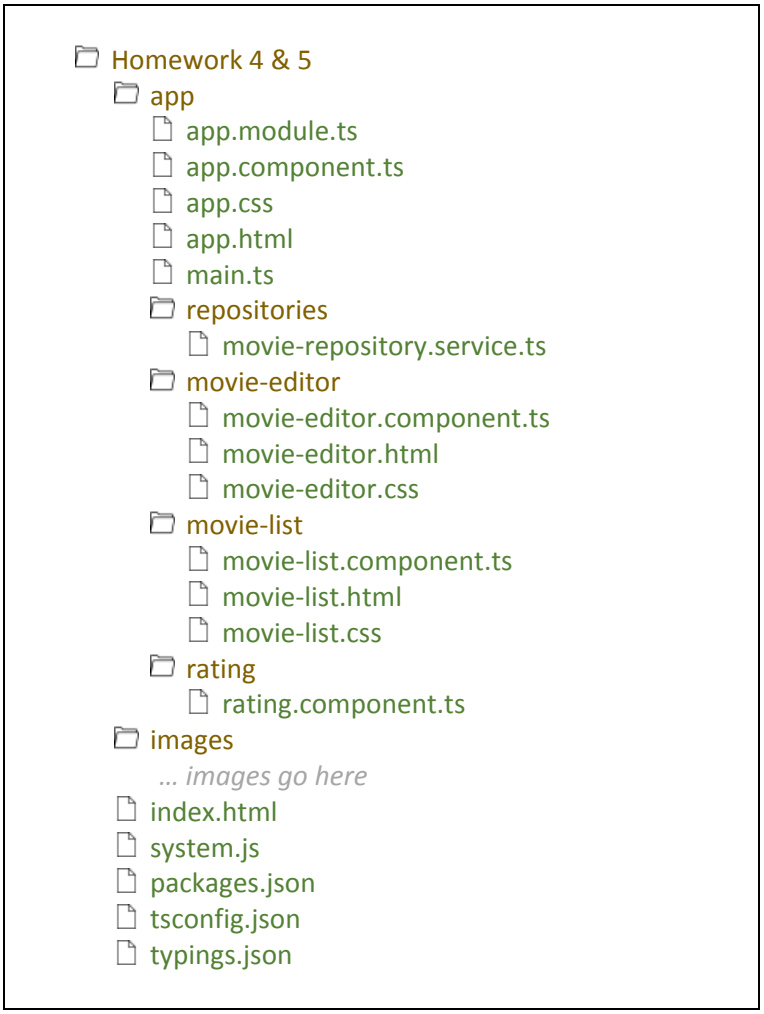
Email to me in a zip file with your name in the file name.

Do **NOT** include the following items in your submission

- **node_modules** directory
- **typings** directory
- **.cmd** or other executable files

Requirements

1. Upon completion, you should have the following file structure:



2. Your project should have **THREE** routes as follows:

Path	Component	Notes
/	MovieListComponent	
/add	MovieEditorComponent	for adding a new movie
/edit/:id	MovieEditorComponent	for editing an existing movie based on id

3. All components should interact with the appropriate methods in the **MoviesRepository** service which I have stubbed out for you.
4. Movie List:
   - On initial load, displays default list of movies
   - **Add movie** button is shown in the top, right hand corner
   - When **add movie** is clicked, user is redirected to editor
   - The movie template contains this tag (instead of stars)
     `<rating [model]="movie.rating"></rating>`
   - When a movie has an empty value for **imagePath**, default placeholder image is shown (in images)
   - Each movie has an **edit** and **delete** button shown as follows:
     i. Glyphicons
     ii. Supertext to movie title/year (smaller font-size, slightly elevated)

iii. At least 5px away from nearby text and each other
iv. Partial opacity until hovered over (see video)
v. Delete icon is red

- When **edit** is clicked, user is redirected to editor appropriately populated
- When **delete** is clicked, movie is immediately removed from list
- When no movies remain in list, a message is shown in a well that reads "There are no movies in the database."

5. Movie Editor
   - When editing:
     i. Form is fully populated with correct movie
     ii. Title of movie (before editing) is shown in <h1> tag
   - When adding:
     i. Form fields are empty
     ii. <h1> tag says "New Movie"
   - Form contains **Name**, **Year** and **Image Path** fields with associated labels
   - The **Year** field has an associated datalist to offer user suggestions years
     i. Options in datalist use **ngFor** to span last 50 years, starting with most recent
     Hint: Use JavaScript's built-in **fill** and **map** functions:
     `var years = Array(50).fill(0).map((x,i)=>(new Date().getFullYear()-i));`
   - **Save** and **Return to list** buttons are blocked under form
   - **Save** button returns user to the list with changes reflected (movie added or updated in list)
   - **Return to list** button returns user to list, without saving or reflecting changes in list
6. Rating
   - Abstraction of stars and badge previously part of movie list (with needed adjustments)
   - Takes model as an input from parent component
   - Given its simplicity, you may use **template** instead of **templateUrl** for this component if you wish
   - On initial load, shows correct number of shaded vs unshaded stars and correct numeric rating value
   - Updates star shading and numeric rating value upon clicking a star

Happy Coding! :-D