Objective

Build an Angular application which does the following:
1. Displays a list of message posts
2. Enables a user to log in and submit new posts or edit existing posts

General Requirements

1. Must be an Angular project generated with the Angular CLI
2. Uses standard Bootstrap styles
3. Conforms to the page layouts as shown in the attached wireframes
4. Must not use 3rd party libraries, e.g. underscore
5. Uses the following REST API to obtain user and post data:
   https://jsonplaceholder.typicode.com/posts
   https://jsonplaceholder.typicode.com/users

Application Requirements

Home page

1. When launching the application, the user is sent to the Home page by default
2. If the user is not logged in, show "Log In" button
3. If the user is logged in, show a “Welcome” message with the user's full name
4. Keep the user logged in (across browser sessions) until the user logs out
5. Display a table with the following details:
   a. a User column containing the user’s full name and company name
   b. a Post column containing the post title and post body
   c. show 10 rows per page
   d. a "New Post" button, which is visible after user logs in
   e. a Log Out button, which is visible after the user logs in
   f. a set of left/right arrows (buttons) for paging
6. The Home page should support the following actions:
   a. pressing the Log In button sends the user to the Log In page
   b. clicking on the company name in the User column goes to the company web site
   c. clicking on the right arrow displays the next 10 posts (disabled if no more posts)
   d. clicking on the left arrow displays the previous10 posts (disabled if no previous posts)
   e. in logged in state, New Post and Log Out buttons are shown
      i. New Post button navigates to Post Edit page in empty state
      ii. Log Out button kills user session and refreshes Home page
7. in logged in state, clicking on post title navigates to Post Edit page in edit state
   a. The user should only be able to edit their own posts

Login page

1. Contains the following fields:
    a. User Name field
    b. Log In button
2. Log In button should be disabled until the user name is entered
3. If the user name does not match a user from the /users endpoint, display an error message
4. If the user name matches a user from the /users endpoint, send user back to Home page in logged in state

Post Edit page

1. The same form should be used for submitting new posts, as well as editing existing posts
2. The form contains the following elements:
    a. Title and Message fields
    b. Delete button (in edit state only)
    c. Save/Cancel button
    d. “Go back to Home page” link (displayed in header)
3. The form should enforce the following validation rules:
    a. the Title and Message fields cannot be empty
    b. title cannot be longer than 200 characters
    c. message cannot be longer than 2000 characters
    d. validation errors should appear below the field that is in error
    e. Save and Cancel buttons should be enabled only when form has been modified and has no validation errors
    f. if the user attempts to navigate away from the page with unsaved data (for example, when clicking on “Go back to home page” header), display a confirmation dialog and allow the user to remain on the page
4. The form should support the following actions (NOTE: the REST endpoint **does not** actually post or update a resource, so the data must be managed locally, i.e. Posts should be added/updated/deleted only locally):
    a. when a new post is saved, the user should be sent back to the Home page, with the newest post shown first in the table
    b. when an existing post is updated, the user should be sent back to the Home page, with a message indicating that the post was saved
    c. when a post is deleted, show a confirmation window, e.g. “Are you sure you want to delete this post?”

Wireframes

Refer to the attached wireframes for guidance on page layouts