**Task 5: Modules & Services**

Courses page implementation continue

Create service for courses. Implement next methods:

1. Get list
2. Create course
3. Get item by id
4. Update item
5. Remove item

Do not use http module to solve a task.

Implement static collection of courses in the service.

Remove static array in your courses component. Use service instead. Call service for items in appropriate lifecycle hook.

Implement delete action.

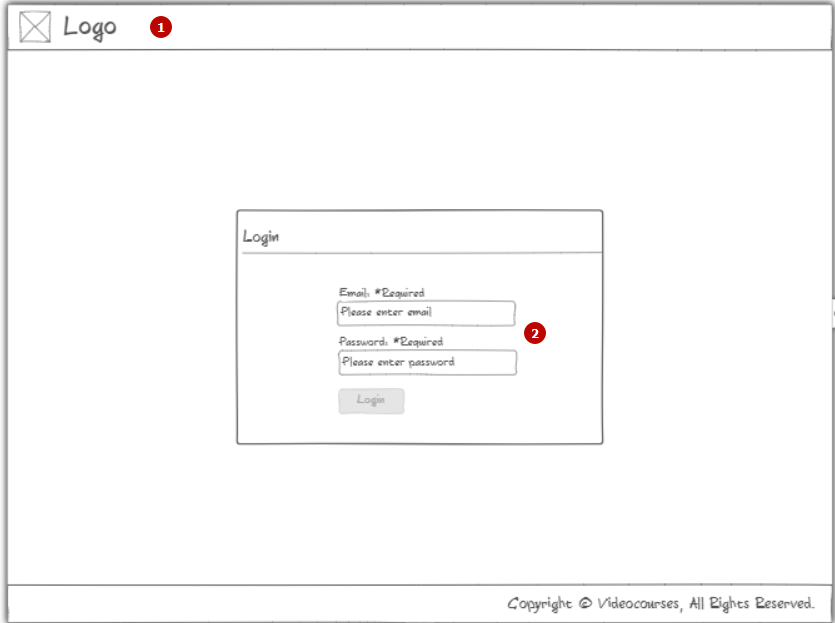
Add confirmation modal on delete item. (Do you really want to delete this course? Yes/No). You can use simple prompt window, third party package, your own solution to render modal.

Create authorization service. Implement next methods:

1. Login (stores fake user info and token to local storage)
2. Logout (wipes fake user info and token from local storage)
3. IsAuthenticated (boolean)
4. GetUserInfo (returns user login)

Use authService in header component. Implement fake action on logout (wipe user info, console.log action name)

Implement Login page:



1. Reuse header/footer components. Implement logic to hide breadcrumbs and user info block (call isAuth and act corresponding to result. Use \*ngIf.
2. Create markup, module and component for login page. Do not implement any validation here. No form for this step. Just bind value to the component class property.

On “Login” button click - use login method of your authService. Then just console log ‘logged in successfully’. Fake for now.

You don’t have to make routing. At this step we don’t know what routing is. So please use familiar construction below:

Place <login-page></login-page> into app component.  
\*If you want to use routing - please go ahead. You don’t need but no restriction here.

Criteria for evaluation

1. Nothing changed
2. Added some simple markup + css for login page. Added services but not working.
3. Partially done with critical issues
4. Added implementations for services, login page.
5. Done all above without major issues

Extra task:

Proceed with unit testing. Now you have your first services. It’s time to cover them with unit tests. Related materials provided in the micro lecture after general.