# Ads – AngularJS Practical Project

You are assigned to design and implement a web site for **Online Ads Publishing** as **single page application (SPA)** using **HTML5** and **AngularJS**. The app manages users and their ads organized by towns and categories. Anonymous site visitors can view published ads. Users can register, login, view their ads, add, edit and delete ads and logout. Administrators approve ads before publishing and can manage the users, ads, categories and towns. You are given the server-side REST services to be called by your app with AJAX requests so you do not need to develop back-end.

All application screens are provided as UI prototype: [Ads-Project-AngularJS-Screens.pdf](file:///C:\Software-University\Courses\AngularJS\December-2014\Ads-Project-AngularJS-Screens.pdf).

## Important: Commit Every Day in GitHub

Please **use GitHub** for your project development!

* Commit several times a day.
  + You need to show many small commits that indicate your constant work on the project.
  + Avoid committing large blocks of code at once.
* You should prove that you have worked at least 3-4 days over your project!

Note that Git does not require Internet connection in order to commit changes. You can commit locally and push your changes to GitHub at once. Please commit many times to show your work progress step by step.

Up to 100 score

## Ads REST Services

You are given the following REST services for your Ads SPA application.

* Services base URL (in the Windows Azure cloud): <http://softuni-ads.azurewebsites.net/api/>
* Services help page (online documentation): <http://softuni-ads.azurewebsites.net/Help>
* Services source code: <https://github.com/SoftUni/SPA-with-AngularJS/tree/master/Ads-REST-Services>
* Postman sample HTTP requests for each service: <https://github.com/SoftUni/SPA-with-AngularJS/blob/master/Ads-REST-Services/Ads-REST-Services.json.postman_collection>

The documentation below provide additional details for the Ads REST services.

## Public Services

Public services are accessible for site visitors without login:

* **Get All Published Ads**
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/ads>, Method: GET
  + URL parameters:
    - CategoryId (optional) – filter ads by category
    - TownId (optional) – filter ads by town
    - PageSize (optional) – ads per page, e.g. 5 (range: 1…1000, default value 10)
    - StartPage (optional) – a starting page (1 … number of pages)
  + Returns (JSON): {"numItems":…, "numPages":…, "ads":[ad, ad, …]}
    - numItems – the total number of items (in all pages)
    - numPages – the number of pages the items are split into, typically calculated by the formula **(numItems** **+** **PageSize** **-** **1)** **/** **PageSize**
    - ads – an array of items in the current page (**0** **…** PageSize items)
* **Get All Categories**
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/categories>, Method: GET
* **Get All Towns**
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/towns>, Method: **GET**
* **User Login**
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/user/login>, Method: POST
  + Parameters: username, password
  + Returns (JSON): {"access\_token":…, …}
    - If the logged-in user is administrator, the returned JSON holds "isAdmin":"true"
  + The returned access token is used for authorization in all further requests to the service (should be passed as HTTP header "Authorization: Bearer access\_token")
* **Register User**
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/user/register>, Method: POST
  + Parameters: username, password, confirmPassword, name, email, phone, townId (optional)
  + Returns (JSON): {"access\_token":…, …} (just like the login sevice)

## User Services

User services are accessible for successfully authorized users (after login):

* **Create New Ad**
  + Creates a new ad owned by the currently logged in user. The new ad status is WaitingApproval.
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/user/ads>, Method: POST
  + Parameters: title, text, imageDataUrl (optional), categoryId (optional), townId (optional)
  + Headers: Authorization: Bearer user\_access\_token
* **Get User Ads**
  + Returns the ads of the currently logged in user (sorted by date in decreasing order)
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/user/ads>, Method: GET
  + Headers: Authorization: Bearer user\_access\_token
  + Parameters: status (Inactive / WaitingApproval / Published), startPage, pageSize
  + Returns (JSON): {"numItems":…, "numPages":…, "ads":[ad, ad, …]}
* **Deactivate User Ad**
  + Endpoint: [http://softuni-ads.azurewebsites.net/api/user/ads/deactivate/{id}](http://softuni-ads.azurewebsites.net/api/user/ads/deactivate/%7bid%7d), Method: PUT
  + Headers: Authorization: Bearer user\_access\_token
* **Publish Again User Ad**
  + Endpoint: [http://softuni-ads.azurewebsites.net/api/user/ads/publishagain/{id}](http://softuni-ads.azurewebsites.net/api/user/ads/publishagain/%7bid%7d), Method: PUT
  + Headers: Authorization: Bearer user\_access\_token
* **Get User Ad by Id**
  + Endpoint: [http://softuni-ads.azurewebsites.net/api/user/ads/{id}](http://softuni-ads.azurewebsites.net/api/user/ads/%7bid%7d), Method: GET
  + Headers: Authorization: Bearer user\_access\_token
  + Returns (JSON): {"id":…,"title":…,"text":…,"date":…,"imageDataUrl":…,…}
  + The date and time is returned in **ISO-8601** format, e.g. "2014-12-18T00:39:19Z"
* **Edit User Ad**
  + Endpoint: [http://softuni-ads.azurewebsites.net/api/user/ads/{id}](http://softuni-ads.azurewebsites.net/api/user/ads/%7bid%7d), Method: PUT
  + Headers: Authorization: Bearer user\_access\_token
  + Parameters: title, text, changeImage, imageDataUrl, categoryId, townId
  + ImageDataUrl is optional parameter used along with changeImage
    - If you want to leave the image unchanged, send changeImage = false
    - If you want to changed the image, send changeImage = true with new ImageDataUrl
    - If you want to delete the image, send changeImage = true without ImageDataUrl
* **Delete Ad by Id**
  + Endpoint: [http://softuni-ads.azurewebsites.net/api/user/ads/{id}](http://softuni-ads.azurewebsites.net/api/user/ads/%7bid%7d), Method: DELETE
  + Headers: Authorization: Bearer user\_access\_token
* **Change User Password**
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/user/changePassword>, Method: DELETE
  + Headers: Authorization: Bearer user\_access\_token
  + Parameters: oldPassword, newPassword, confirmPassword
* **Get User Profile**
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/user/profile>, Method: GET
  + Headers: Authorization: Bearer user\_access\_token
  + Returns (JSON): {"name":…,"email":…,"phoneNumber":…,"townId":…}
* **Edit User Profile**
  + Endpoint: <http://softuni-ads.azurewebsites.net/api/user/profile>, Method: PUT
  + Headers: Authorization: Bearer user\_access\_token
  + Parameters:name, email, phoneNumber, townId (optional)

After login / register, pass the access session token as HTTP request header to **authorize your requests**:

* Authorization: *Bearer access\_token*

All POST request expect their parameters in application/x-www-form-urlencoded form.

## Administration Services

Administration services are accessible for successfully authorized **administrators** (after login). Administrators are special users who have "administrator" role. Administrators have full access to view, create, edit and delete ads, users, categories and towns.

* Administration REST services are documented here: <http://softuni-ads.azurewebsites.net/Help>.

## Ads Web Design

You аre given the **Web design** and **UI prototype** of the Ads SPA application as PDF document:

* <Ads-Project-AngularJS-Screens.pdf>

You need to convert the design to HTML + CSS. Pixel-perfect layout is not required. You do not need to match exactly the sizes, fonts and colors of the elements. A responsive design is highly desirable. You can use responsive CSS frameworks like Bootstrap.

10 score

## Ads SPA Application

Design and implement a client-side SPA application based on AngularJS for managing the ads.

### AngularJS Project Structure

* Prepare an AngularJS project structure following the industry best practices.
* You should have separate folders for controllers, directives, filters, services, views, etc.

5 score

### Public Screens

Public screens are accessible for site visitors without login.

* **Home Screen**
  + Route: #/
  + **Lists all ads** (without login).

5 score

* + Implement **paging**.

5 score

* + Implement **filtering by category and town**.

5 score

* **Login Screen**
  + Route: #/login
  + **Logins an existing user**. Shows notification for success or error message.
  + After login, the user is automatically redirected to the user home screen.

5 score

* **Register User Screen**
  + Route: #/register
  + **Registers a new user**. Shows notification for success or error message.
  + **After registration, the user** is automatically logged in and is redirected to the user home screen.

10 score

### User Screens

User screens are accessible for authorized users only (after login).

* **User Home Screen**
  + Route: #/user/home
  + This screen is the same like the Home screen, but includes a user **navigation sidebar** at the left.

5 score

* **Publish New Ad Screen**
  + Route: #/user/ads/publish
  + **Creates a new ad** in the system and submits it for publishing.

10 score

* + Implement "**Choose Image**" functionality with live image preview.

5 score

* **List My Ads Screen**
  + Route: #/user/ads
  + **Lists user's ads**.

5 score

* + Show [Deactivate], [Publish Again], [Edit] and [Delete] **buttons** depending on the ad status.

5 score

* + Implement **paging**.

5 score

* + Implement **filtering by status** (Published / Waiting Approval / Inactive).

5 score

* **Deactivate / Publish Again Ad**
  + The [**Deactivate**] button should make an ad inactive. Available for ads in state Published and Waiting Approval.
  + The [**Publish Again**] button should submit an ad for approval. Available for inactive ads only.
  + Shows notification for success or error message.

5 score

* **Edit Ad**
  + Route: #/user/ads/edit/:id
  + Users should be able to **edit their inactive ads**. Show notification for success or error message.

10 score

* + Implement "**Change Image**" and "**Delete Image**" functionality with live image preview.

5 score

* **Delete Ad**
  + Route: #/user/ads/delete/:id
  + Users should be able to **delete their inactive ads**. Show notification for success or error message.

5 score

* **Edit User Profile**
  + Route: #/user/profile
  + Users should be able to **edit their profile data**. Show notification for success or error message.

5 score

* **Change User Password**
  + Route: #/user/profile
  + Users should be able to change their password. Show notification for success or error message.

5 score

* **Logout**
  + Route: #/logout
  + Successfully logged in users should be able to **logout** from the app.
  + Logout shows a notification message and redirects to the Home screen.

5 score

* **Authorization Checks**
  + Anonymous site visitors (without login) should be able to access the Public Screens but should not be able to access User Screens and Administration Screens.
  + An attempt to access anonymously these screens should redirect the user to the Home screen.

5 score

### \* Bonus: Administrator Screens

Implement the Administrator Screens and functionality as bonus.

Administrator screens are accessible for users who are administrators (after login):

* **Admin Home Screen** – **#/admin/home**
* **Admin Approve Ad**
* **Admin Reject Ad**
* **Admin Edit Ad** – **#/admin/ads/edit/:id**
* **Admin Delete Ad** – **#/admin/ads/delete/:id**
* **Admin List Users** – **#/admin/users/list**
* **Admin Edit User** – **#/admin/users/edit/:id**
* **Admin Change User Password** – **#/admin/users/edit/:id**
* **Admin Delete User** – **#/admin/users/delete/:id**
* **Admin List Categories** – **#/admin/categories/list**
* **Admin Create Category** – **#/admin/categories/create**
* **Admin Edit Category** – **#/admin/categories/edit/:id**
* **Admin Delete Category** – **#/admin/categories/delete/:id**
* **Admin List Towns** – **#/admin/towns/list**
* **Admin Create Town** – **#/admin/towns/create**
* **Admin Edit Town** – **#/admin/towns/edit/:id**
* **Admin Delete Town** – **#/admin/towns/delete/:id**

Bonus: up to 200 score