# Angular Exam – Animal Kingdom

# Exam rules:

* You have 6 hours – from 15:00 to 21:00
* When you are ready, delete the *node\_modules* folder, make sure all dependencies are listed in the *package.json* file and submit your archived project at <https://softuni.bg/trainings/1644/angular-2-fundamentals-july-2017>
* There will be no breaks during the exam but you can go to the toilet or to breathe some fresh air outside **if you are alone**
* Good design is optional. Make sure the site is usable enough but do not spent too much time on HTML & CSS.
* If you have any questions for the following description – ask the trainer, he is somewhere in the room. ☺
* Have fun! ☺

Open the server directory, install all its dependencies and run it. It is listening on port 5000. The server uses in-memory database so if you need to restart it for some reason, all the data will be lost and you will need to populate it again.

## Create an Angular application and add authentication

Create an Angular application and prepare the initial project structure. Install Redux if you prefer (it is not necessary, you may skip it and create pure Angular application). Add authentication and make sure the register, login and logout functionalities work correctly. To register a user, you need to send a POST request to the server on ‘**/auth/signup**’ with ‘name’, ‘email’ and ‘password’ data (sent as JSON). To login a user, you need to send a POST request to the server on ‘**/auth/login**’ with ‘email’ and ‘password’ data (sent as JSON). You need to save the user token in your application state. Make sure you validate everything on the client application.

## Add statistics (10 points)

Show the total number of users and animals in the system on the home page. You need to make a GET request to ‘**/stats**’ in order to retrieve the data.

## Add creating of animals (10 points)

Add a form to create animals in the system. Each animal has ‘name’ as string, ‘age’ as number, ‘color’ as string, ‘type’ as string (type can be ‘Cat’, ‘Dog’, ‘Bunny’, ‘Exotic’ or ‘Other’), ‘price’ as number, ‘image’ as string URL and optional ‘breed’ as string. Make sure you validate everything on the client application. The data must be sent as POST request to the server on ‘**/animals/create/**’. This route is only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks.

## Add listing of animals (10 points)

Add a page where all animals are listed. Since the data from the server comes in pages of 10 animals, your page must have buttons to navigate between the pages. Try to validate the buttons when the page is no longer valid. You need to make a GET request to ‘**/animals/all**’ to receive an array of animals data. Optionally, you can pass a query string parameter ‘page’ to request more data, for example ‘**/animals/all?page=2**’. Link each animal to its details page. Don’t show every piece of information about the animal on this page. Leave something for the details page. You may add this functionality on the home page.

## Searching for animals (10 points)

Add an option to search for animals on the listing page. Add a text input and a button. You need to make a GET request to ‘**/animals/all**’ with a query string parameter ‘search’ to retrieve the search results, for example ‘**/animals/all?search=Malcho**’. Keep in mind that the ‘page’ and ‘search’ parameters can be combined. For example, the request ‘**/animals/all?page=3&search=Myrkins**’ will return the third page of the searched term ‘Myrkins’.

## Add animal details (10 points)

Add a page where all animal details are shown. You need to make a GET request to ‘**/animals/details/{*id*}**’ to retrieve information about the animal with the provided id. This route is only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks. Make sure your Angular application redirects to the login page, if the user tries to open the animal details page and she’s not logged in.

## Add option to add reactions and comments (15 points)

On the animal details page add an option for the user to put a reaction for the current animal. Reactions have ‘type’ – ‘like’, ‘love’, ‘haha’, ‘wow’, ‘sad’, ‘angry’. You need to make a POST request to ‘**/animals/details/{*id*}/reaction’** in order to create a reaction. The id is the animal id the user is reacting on. Additionally, add a comment form, which makes a POST request to ‘**/animals/details/{*id*}/comments/create**’. Each comment has ‘message’ as string. Make sure you validate everything on the client application. Both routes are only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks.

## Add listing of all comments (10 points)

On the animal details page, you need to list all comments for the current animal. You need to make a GET request to ‘**/animals/details/{*id*}/comments**’ to receive an array of comments data. This route is only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks.

## Add profile page (10 points)

Add a profile page where all animals created by the current user are shown. You need to make a GET request to ‘**/animals/mine/**’ to receive an array of animals. This request does not have paging so you do not have to implement it. The route is only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks.

## Add option to delete animals (15 points)

On the profile page, for each animal add a delete button to allow the user to remove a previously added animal from the system. To delete an animal, you need to make a POST request to ‘**/animals/delete/{*id*}**’. This route is only for authenticated users so you need to send a header with `Authorization` name and value `bearer {*token*}` in order to pass the authentication checks.