

JS Back-End Exam – Wildlife Photography

1. Exam rules:

1. You have **4 hours**
2. When you are ready, delete the **node_modules** folder, make sure all dependencies are listed in the **package.json** file and submit your archiving project.
3. You are provided with **HTML & CSS** resources
4. You may **add attributes** (such as **class** and **dataset**), but it is forbidden to **change existing** attributes (such as **class** and **id**)
5. You may **change "href"** attributes on links and add/change the **method** and **action** attributes of HTML Forms.
6. Use **Express.js** as a back-end framework
7. Use **MongoDB** as a database with **mongoose**
8. You can use whatever **view engine** you like (**express-handlebars**, EJS, Pug, etc....)
9. Use **bcrypt** for hashing the password
10. The application **must start** from file **"index.js"** on port **3000**
11. It is **forbidden** to use **React, Vue, Angular**, etc.

2. Application Overview

Get acquainted with the provided HTML and CSS and create an application for sharing posts of **wildlife**.

The visitors can view the **Home page** and **All posts** for wildlife, they can also **register** with **first, last name, email**, and **password**, which will allow them to create their **posts**, to **vote on posts** (if the **current user** is **not** the **author of the post**). Authors can **edit** or **delete** posts at any time.

3. Functional Requirements

The **Functional Requirements** describe the functionality that the **application** must support.

Guest (not logged in)

Guest navigation example:



The **application** should provide **Guest** (not logged in) users with the functionality to **login**, **register**, **view the Home page**, **All posts** page, and the **Details** page.

Users (logged in)

User navigation example:



The **application** should provide **Users** (logged in) with the functionality to:

- **View Home page and all other pages with logged-in navigation**
- **View All posts**

- Create new post [Create post]
- Access post details page [Details]
- Voting on posts (if the **current user** is **not** the author on the post)
- Delete or Edit post depending on user's authentication (**only for the author of the current post**)

4. Database Models

The **Database** of the **Wildlife Photography** application needs to support **2 entities**

User

- First Name - string (required),
- Last Name - string (required),
- Email - string (required),
- Password - string (required),
- My Posts - a collection of Post (a reference to the Post Model)

Note: When a user creates a new post, a reference to that post is added to that collection (**My Posts**).

Post

- Title - string (required),
- Keyword - string (required),
- Location - string (required),
- Date of creation - string (required),
- Image - string (required),
- Description - string (required),
- Author - object Id (a reference to the User model),
- Votes on post - a collection of Users (a reference to the User model),
- Rating of post - number, default value 0

Note: When a user votes on a given post, their **id** is added to that collection (**Votes on post**).

Rating of post will store the overall rating of votes. The value is changing depending on the type of vote (positive and negative).

Implement the entities with the **correct data types**.

5. Application Pages (80 pts)

Note: Don't forget to change the paths to the CSS files, and the images.

Home Page (logged out user)

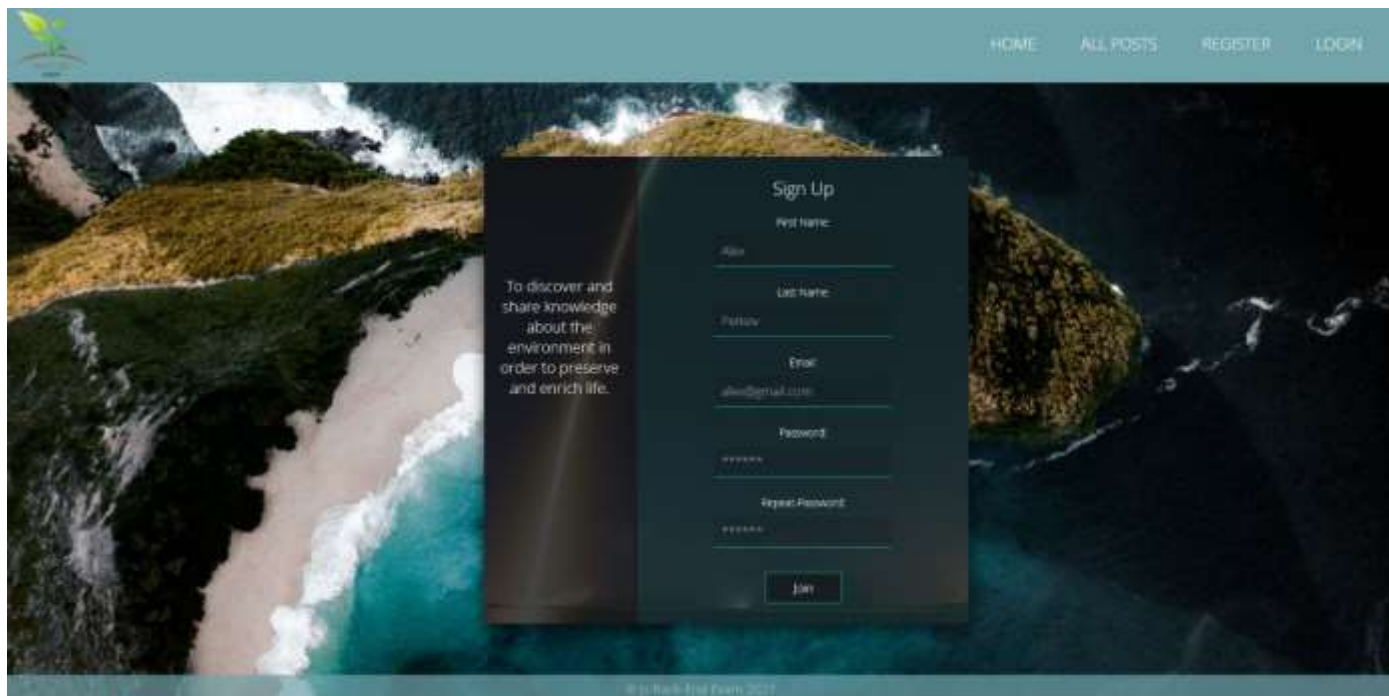


Home Page (logged in user)



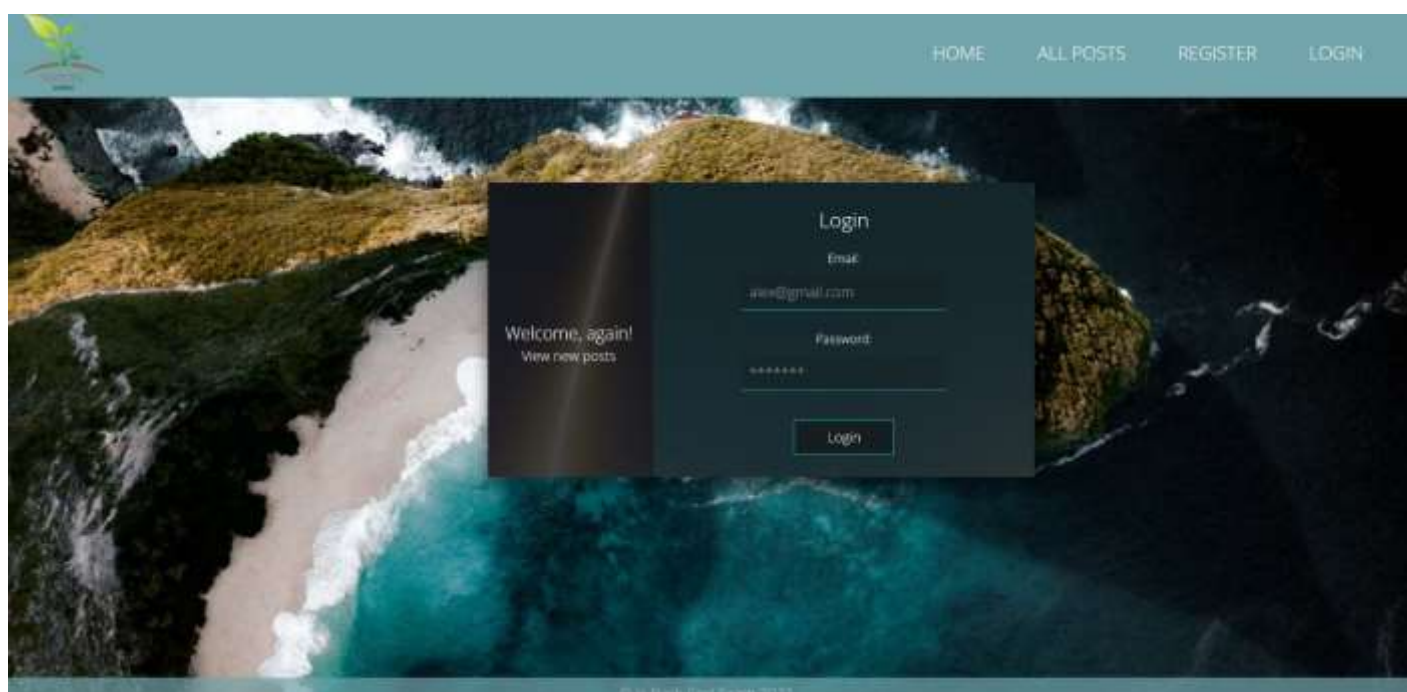
Register Page (logged out user)

Register a user inside the database with **first**, **last name**, **email**, and **password**. **Password** inside the database must be **hashed** (use bcrypt) and both **passwords** must **match**! After successful registration **redirects to the Home page**, with an already logged-in user.



Login Page (logged out user)

Logging an already registered user with the correct **email** and **password**. After successful login **redirects to the Home page**, with an already logged-in user.



Logout (logged in user)

The logout action is available to **logged-in** users. Upon success, clear any session information and **redirect** the user to the **Home** page.

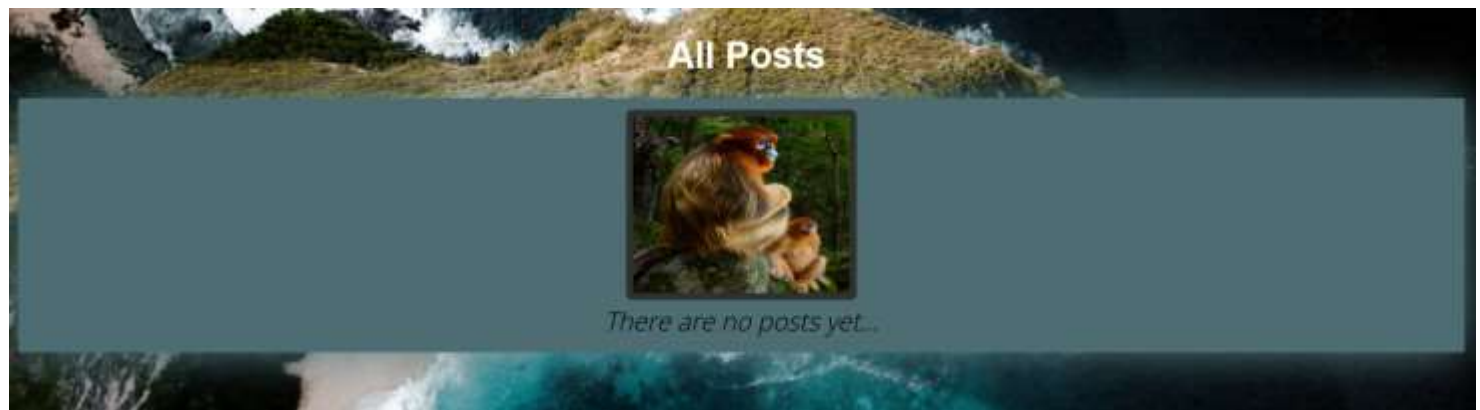
All posts (for logged in users and logged out users)

List of all **wildlife posts**. Each post should be showing information about the **wildlife image**, **keyword**, **title**, **description**, as well as a page with **details** about the **specific post** (the **title**, **description**, and **[Detail]** button will be visible when the **mouse cursor** reaches the image). Like in the picture below:



[Details] button should be a link to the **details page** for the current post.

If there are no wildlife posts in the **database**, display the following view:



Details Page - (for logged in users and logged out users)

All users should be able to **view details** about the post. Clicking the **Details** button in a **post card** should **display** the **Details** page. If the currently **logged-in user is the author** of the post, the **Edit** and **Delete** buttons should be displayed, otherwise they should not be available.

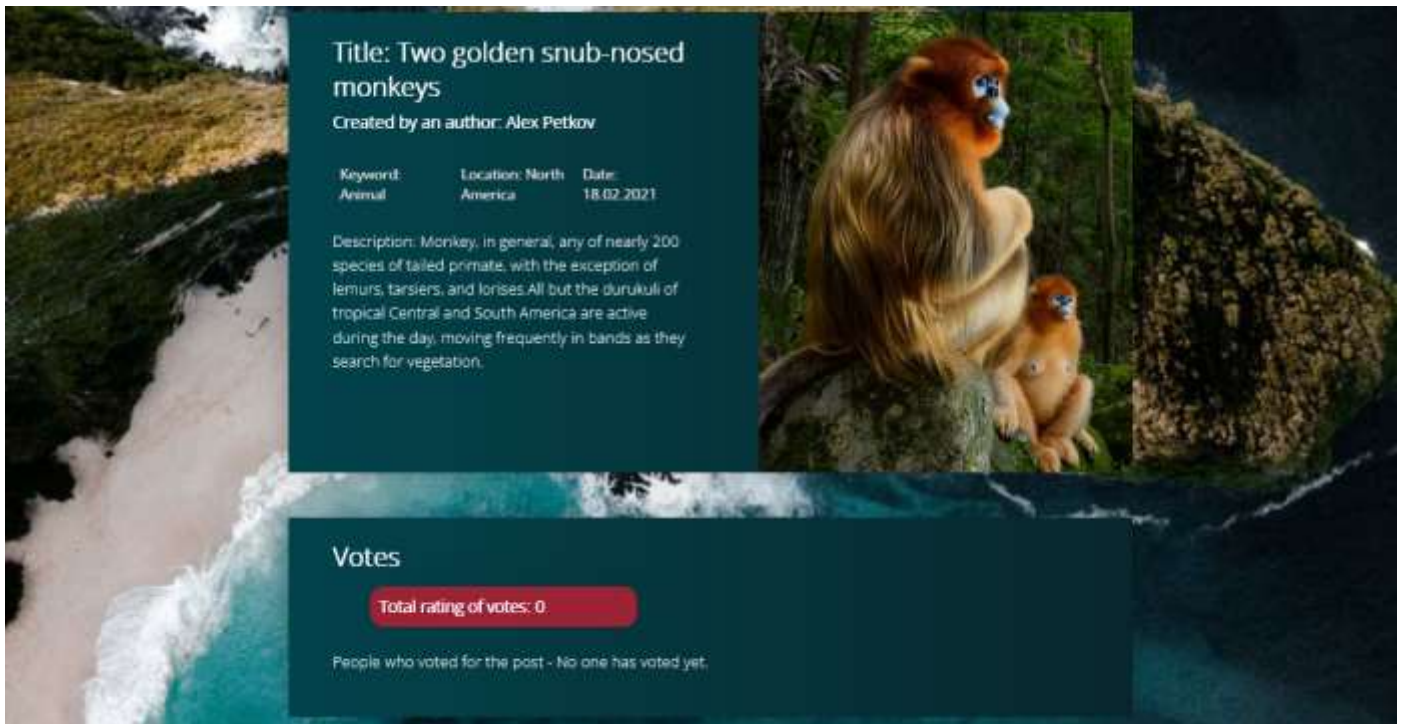
Information about the post:

- **Title**
- **Author**
- **Keyword**
- **Location**
- **Date**
- **Wildlife Image**
- **Description**
- **Buttons** (Depending on the status of the currently logged in user)
- **Votes for this post**
 - Total rating of votes
 - If there are people who voted, separate their emails with **comma** and **space** ", "

- If not, display "**No one has voted yet**".

Details Page (logged out users)

If there are **no logged-in** users, **no buttons** should be displayed.



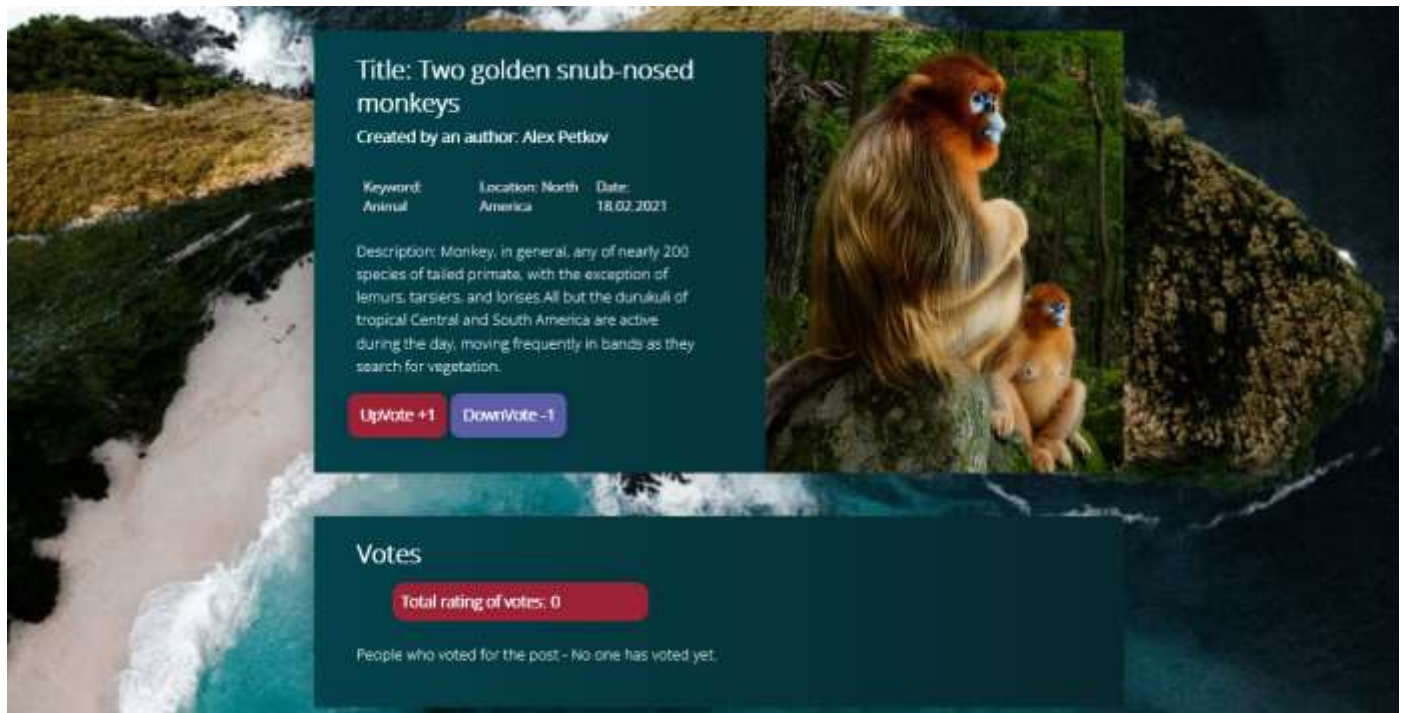
Details Page (logged in user and author of the current post)

If the **currently logged-in** user is the **author** (the user who **created the wildlife post**), he should see the **[Edit]** and **[Delete]** buttons.



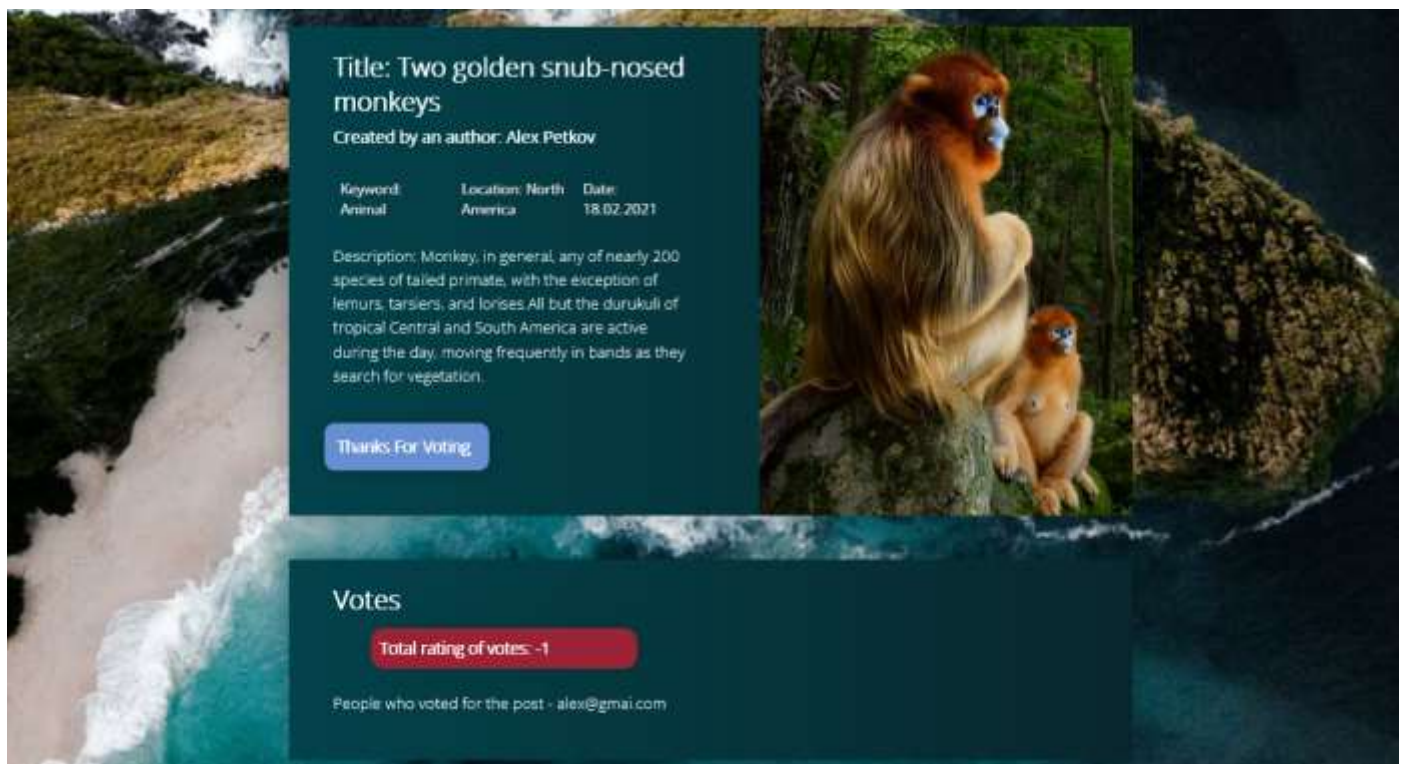
Details Page (logged in user who did not vote)

If the currently logged-in user is **not the author** (the user that is not the creator for that post and has not yet given his vote), he should see the **[UpVote +1]** and **[DownVote -1]** buttons.



Details Page (logged in and already voted on this post)

If the currently logged-in user is not **the author** and has **already voted** for the current post, he should see the button **[Thanks For Voting]**.



Vote on the post (logged in user who is not the author of the post)

Any registered user who is not the **author** of the **wildlife post** must be able to vote (if they have not yet voted).

If he succeeds in voting, his **userId** must be added to the collection of **Votes on post**.

We have two buttons available:

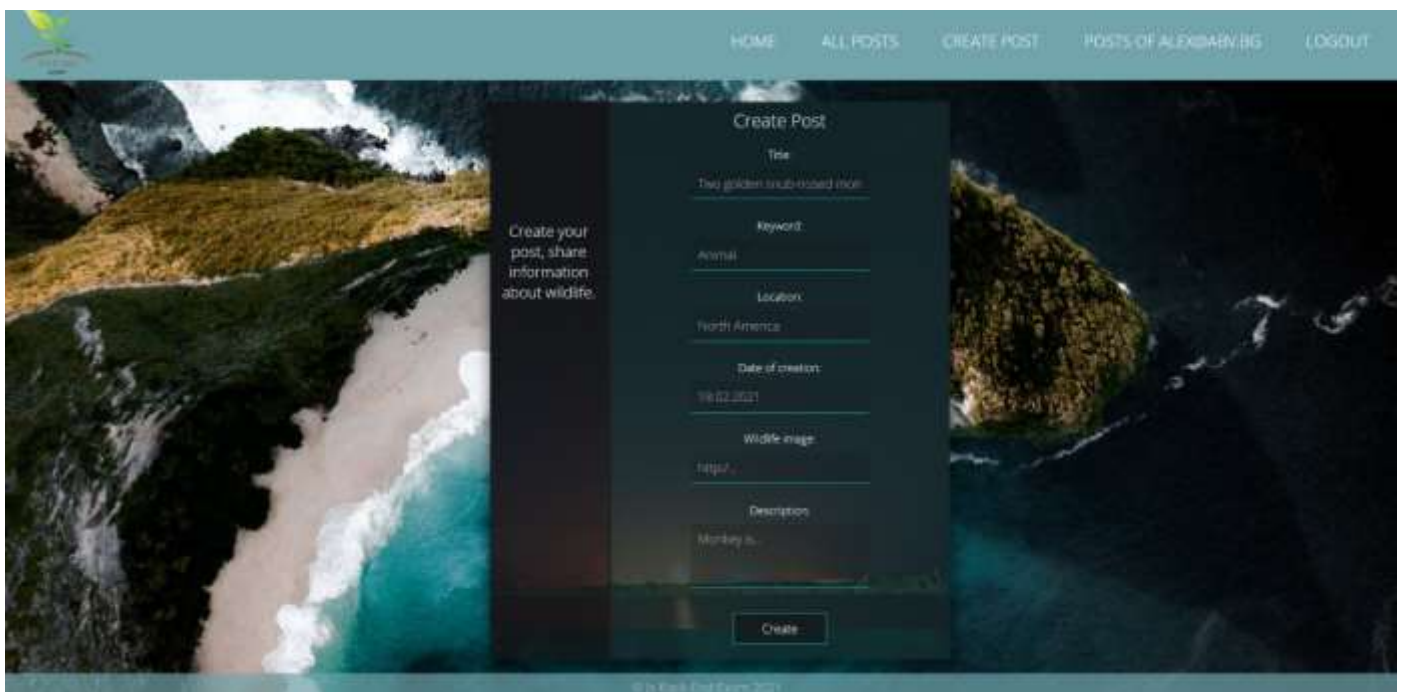
- **[UpVote +1] button** -> The **value** of the **Rating of post** should be **increased by 1**, and the changes should be reflected in the **Total rating of votes**.
- **[DownVote -1] button** -> The **value** of the **Rating of post** should be **reduced by 1**, and the changes should be reflected in the **Total rating of votes**.
- Then redirect the user to the **details page** of the current **wildlife post**.

In the list- **People who voted for the post**, the **emails** of the people who voted must be displayed.

If a user has once voted on the post, he should see "**Thanks For Voting**" and in the list- **People who voted for the post**, his email should be displayed.

Create Post Page (logged in user)

The Create page is available to **logged-in users**. It contains a form for adding new wildlife posts. Upon success, **redirect** the user to the **All Posts** page.



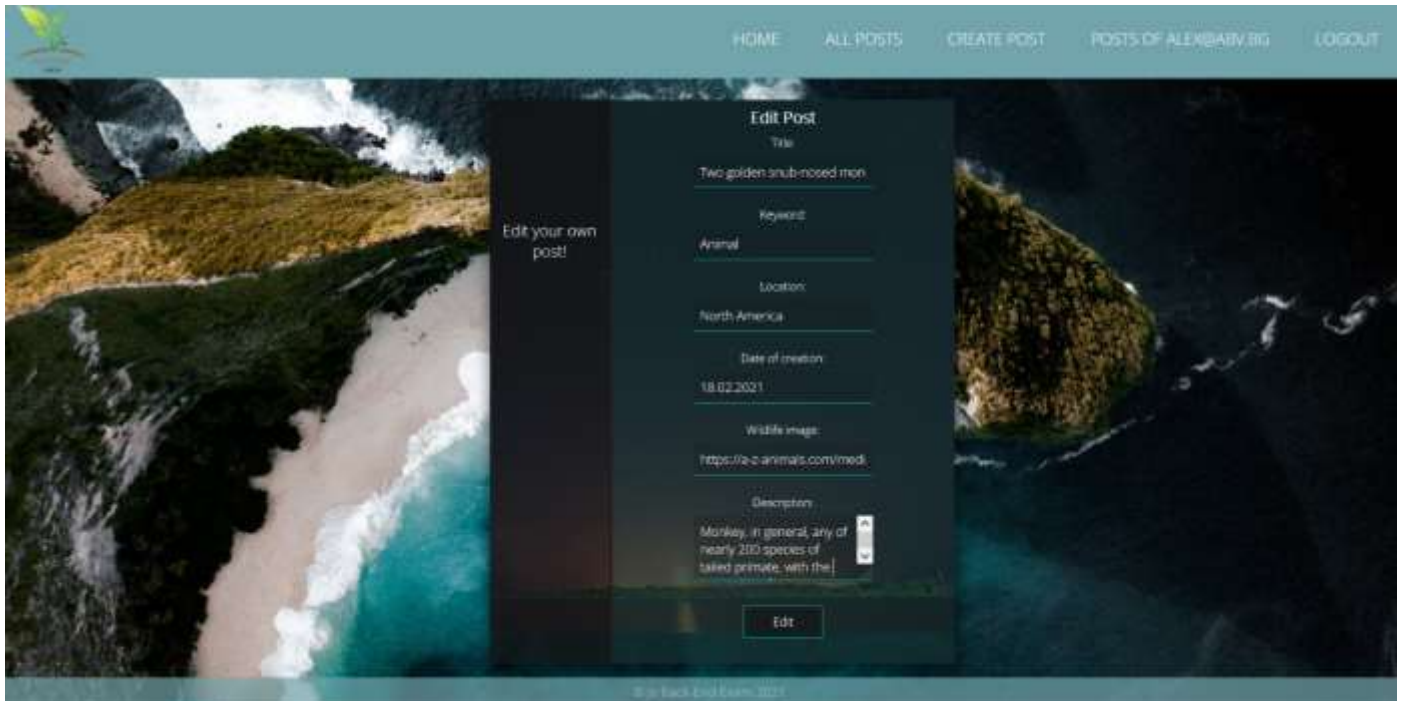
Delete Post (logged in user and author of the current post)

Every author should be able to click over the **[Delete] button** - deleting the current post from the **database** and the user should be **redirected** to the **All post** page.

Edit Post (logged in user and author of the current post)

The **Edit** page is available to logged-in users and it allows authors to **edit** their posts. Clicking the **[Edit] button** of a particular post on the **Details** page should display the **Edit** page, with **all fields filled** with the data for the post. It

contains a form with input fields for all relevant properties. Upon success, **redirect** the user to the **Details** page for the current post.



6. Security Requirements (Routes Guards) - (10 pts)

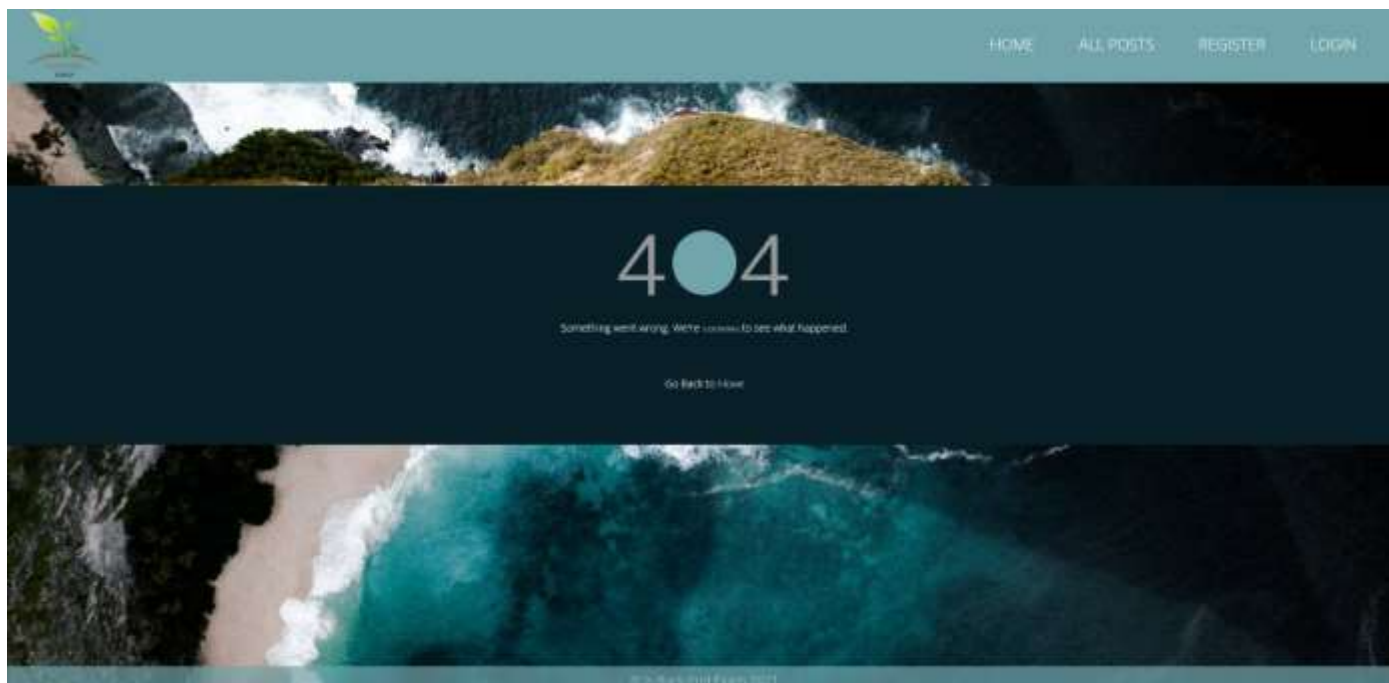
The **Security Requirements** are mainly **access** requirements. Configurations about which users **can access** specific functionalities and pages.

- **Guest** (not logged in) users can access the **Home** page.
- **Guest** (not logged in) users can access the **Login** page and functionality.
- **Guest** (not logged in) users can access the **Register** page and functionality.
- **Guest** (not logged in) and **Users** (logged in) can access **All posts (Listed all posts)**.
- **Guest** (not logged in) can access the **Details** page without functionality.
- **Users** (logged in) can access the **Home** page.
- **Users** (logged in) can access the **Details** page and functionality.
- **Users** (logged in) can access **Create Post** page and functionality.
- **Users** (logged in) can access to **Vote on posts** functionality.
- **Users** (logged in and **author of the current post**) can access the **Delete and Edit post functionality**.
- **Users** (logged in) can access **Logout** functionality.

If **Guests** (not logged in) trying to **access** a page that it should **not be able to**, you must **redirect** them to the **Login** page.

If **Users** (logged in) trying to **access** a page that it should **not be able to**, you must **redirect** them to the **Home** page.

Use the following view for **invalid paths**:



7. Validation and Error Handling (10 pts)

The application should **notify** the users about the result of their actions.

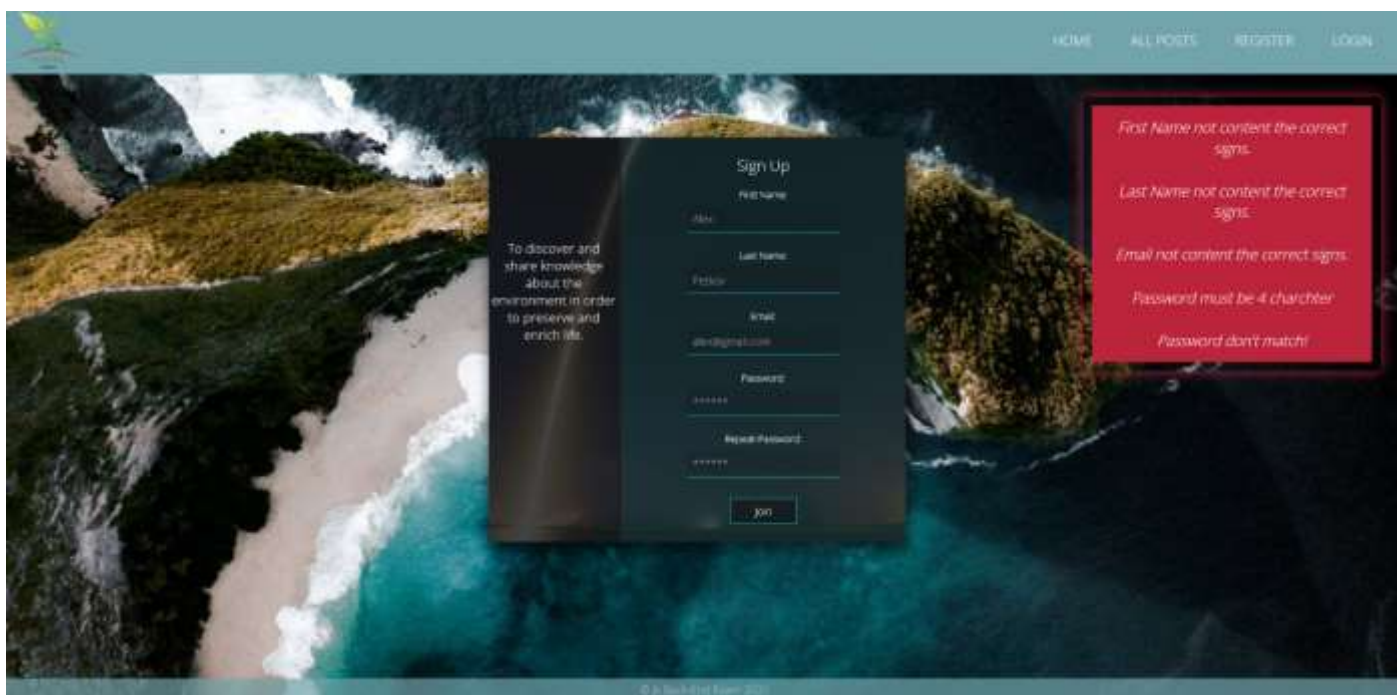
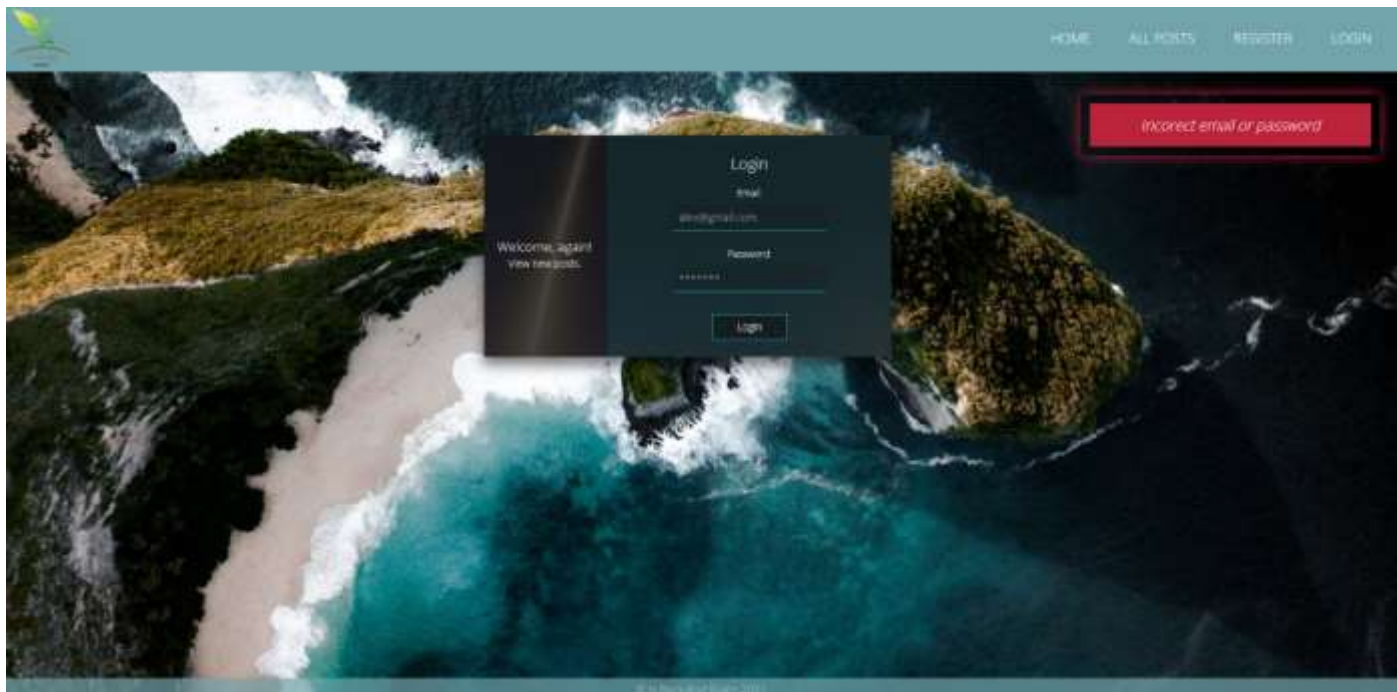
In case of error, you **should display** div with class **"error-container"**.

You can choose to display the first error or all of them. You have complete freedom to choose the content of the error message you will display.

Login / Register

You should make the following validations:

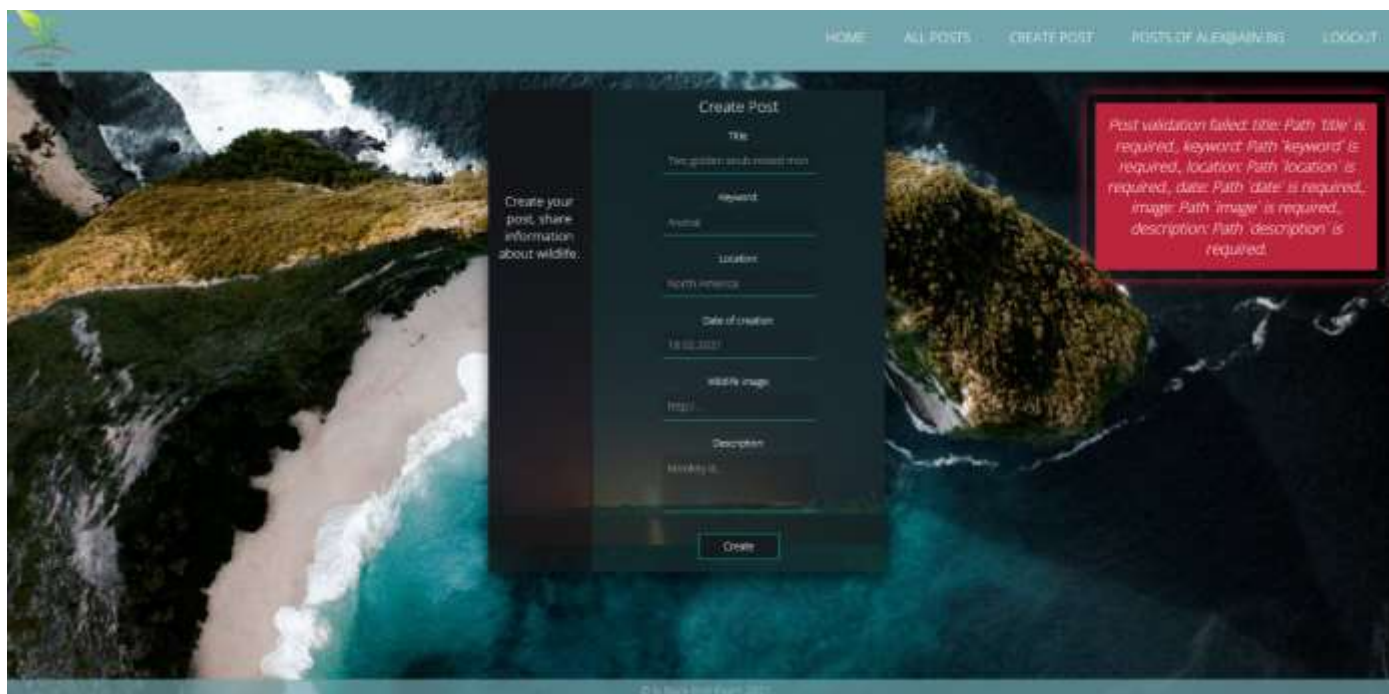
- The **first name** should be **at least 3 characters** long and contains only English letters
- The **last name** should be **at least 5 characters** long and contains only English letters
- The **email** should be **in the following format: <name>@<domain>.<extension>**
 - Only Latin letters are allowed for any of the parts of the email
 - There must be a **point(.)** after the **<domain>**
 - Example of a valid email - **"petar@softuni.bg"**
- The **password** should be **at least 4 characters** long
- The **repeat password** should be **equal to the password**



Post

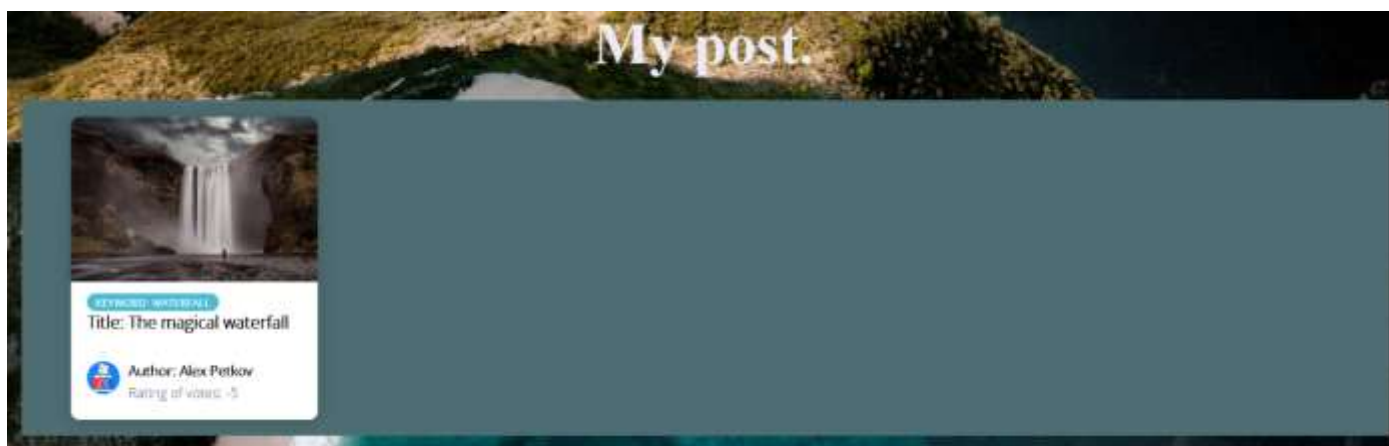
You should make the following validations while **creating and editing** a post:

- The **Title** and **Keyword** should be **at least 6 characters (each)**.
- The **Location** should be a maximum of **10 characters** long.
- The **Date** should be exactly 10 characters - "**02.02.2021**"
- The **Wildlife Image** should start with **http://** or **https://**.
- The **Description** should be a minimum of **8 characters** long.



8. Bonus - My posts page (10 pts)

Each **logged-in user** should be able to view his post by clicking [**Post of {email-of-user}**]. Each **own post** of user should be showing information about the **wildlife image**, **keyword**, **title**, **author** as well a total **rating of votes** for the given post. Like in the picture below:

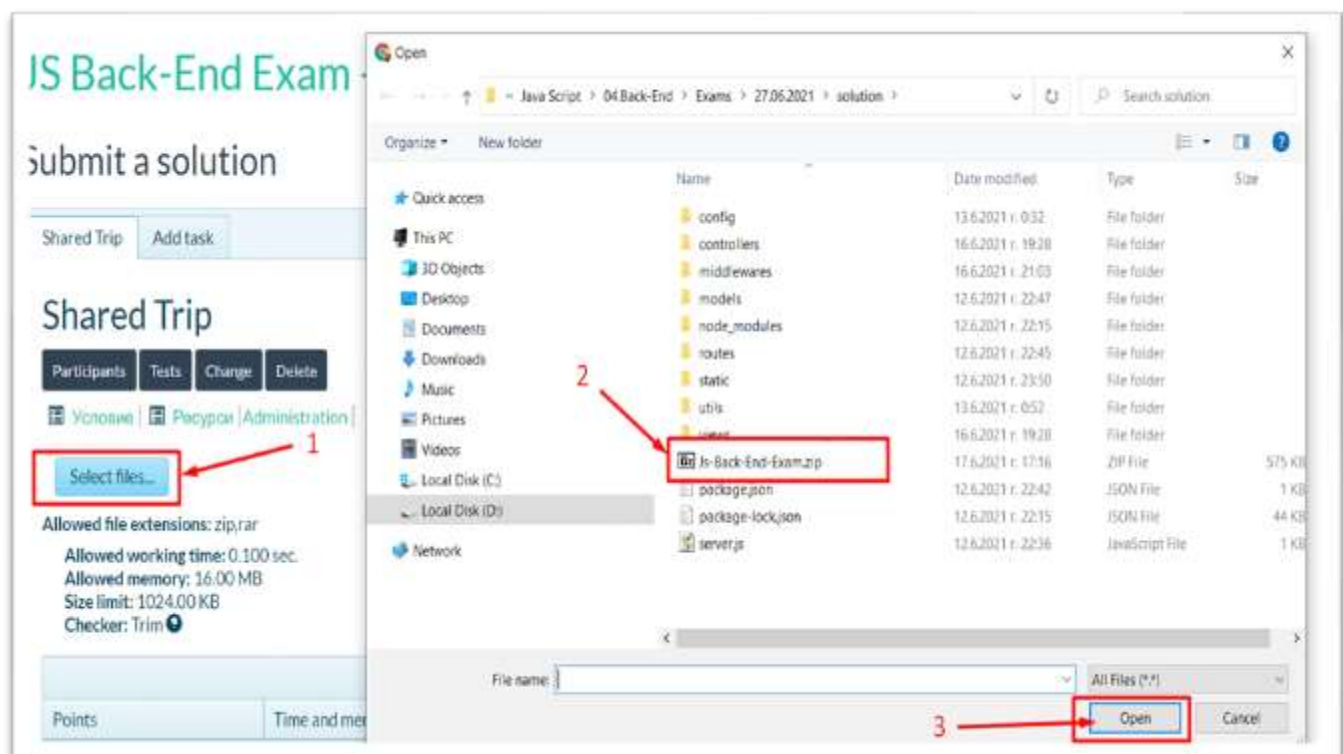
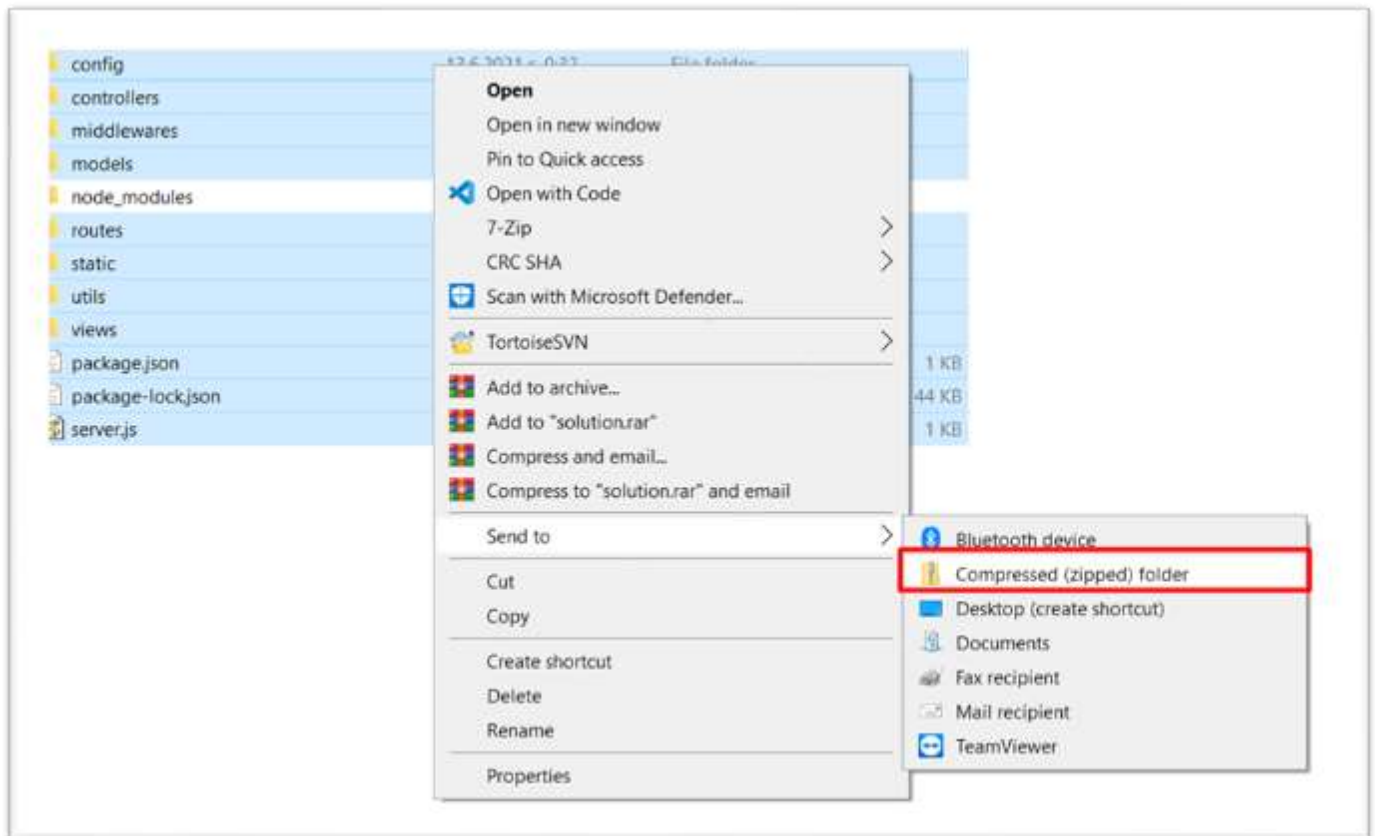


If **there are no own** wildlife posts in the **database**, display the following view:



9. Submitting Your Solution

Place in a **ZIP** file your project folder. Exclude the **node_modules** folder. Upload the archive to Judge.



Shared Trip

Add task

Shared Trip

Participants

Tests

Change

Delete

Условие

Ресурсы

Administration

Select files...

Js-Back-End-Exam.xp

Allowed file extensions: zip,rar
 Allowed working time: 0.100 sec.
 Allowed memory: 16.00 MB
 Size limit: 1024.00 KB
 Checker: Trim

File upload

Submit

Shared Trip

Participants

Tests

Change

Delete

Условие

Ресурсы

Administration

Select files...

Allowed file extensions: zip,rar

Allowed working time: 0.100 sec.

Allowed memory: 16.00 MB

Size limit: 1024.00 KB

Checker: Trim

File upload

Submit

Problem results

| Participant | Result |
|-------------|---------|
| shampion | 0 / 100 |

Submissions

| Points | Time and memory used | Submission date |
|--------------------|----------------------|---------------------|
| Compile time error | | 17:24:04 17.06.2021 |

This is not a problem !

GOOD LUCK! ☺