Express.js Retake – Wiki

Exam rules:

- You have 6 hours – from 9:00 to 15:00
- 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://judge.softuni.bg/Contests/1070/ExpressJS-17-Jun-20178-Exam
- 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**
- **You are required to edit the provided HTML to make it functional and may extend it as needed.**

Application Overview

Get familiar with the provided **HTML & CSS** and create an application that allows users to **create** and **read** encyclopedia **articles**. An anonymous user should be able to access the home page and **view any article**, as well as **register/login**. A registered user can **edit any article**. Your **application** should also have **administrators**. Admins can **lock articles**, so they can't be edited by non-admins.

Since anyone can edit any article, it's important that **edit history** is stored for **each article** and previous versions can be accessed in their entirety. Design your models in such a way as to allow for **multiple older versions** of an article to exist and be viewable.

Problem 1. Home Page & Navigation (20 points)

The **home** page should display all **navigation** menus, an excerpt of the **latest article** (first 50 **words** of the content) and links to the three most **recently added articles**. Registered users see a greeting with option to log out, while anonymous users have the option to login or register (*see provided HTML*).

Make sure only the **necessary** elements are displayed in the navigation menu and on every page, depending on the status of the user (anonymous, logged in, admin).

Problem 2. Create Article (20 points)

For authenticated users

An article must have a **title**, a **locked status** and a list of **edits**. Each edit has an **author**, a **creation date, content**, and an associated **article**. Upon creation of the article, its title is stored, the locked status is set to false and the **contents**, which the user has entered, are **stored in an edit**, associated with that article. The user becomes the author for the edit and the creation date is set to the current time.

Problem 3. List All Articles (10 points)

Show a list of all articles, ordered alphabetically, with a links to display their contents.

Problem 4. Display Article (15 points)

Load the article from the database and display a page, containing its **title** and **content of the most recent edit**. Authenticated users see the option to edit the article.

Hint: For better looking formatting when outputting contents, interpret two line breaks as a new paragraph.

Problem 5. Edit Article (15 points)

For authenticated users

An article can be edited by any registered user. Upon submission, store the contents in an **edit** and associate it with the article. An edit has an **author**, a **creation date, content**, and an associated **article**.

If an article is **locked**, only admins can edit it.

Problem 6. Article History (10 points)

For authenticated users

Show a list with the **edits** for the selected article. Each edit is shown with its **creation date** and **author**. When an edit is selected, a details page with its **contents** is displayed (same as **Display Article**).

Problem 7. Lock/Unlock Article (10 points)

Admin functionality

When an article is locked, only admins can edit it. Regular users can still read it and view its history. When an article is created, it's unlocked by default.

Bonus: Search (10 points)

When a user enters a search criteria, look in the database for **partial matches** and return a list of links to all articles, whose **title contains the input**.