Hand-in assignment 3

Description:

In this assignment, you should write a Single-Page App (SPA) in AngularJS, using a provided REST service (see below). The application should take care of the teaching evaluations at Reykjavik University. There are two users in this application: a) the teaching department (ísl. kennslusvið - called "admin" from here after) and b) students.

Hand-in

Hand in all code required to run your solution in a single archive file (.zip, .rar, .7z) . Make sure NOT to include any node_modules folders in your handin, but ensure it is obvious how to start your project (for instance, by including README.md file in the root of your solution).

Also, ensure that when you hand in your solution, that the database contains at least one valid evaluation template. Ideally, you should structure it in a way you as a student would want teaching evaluations to appear.

Requirements

The following requirements are given.

Functional requirements (50%):

- (5%) The user should be able to log into the application (given a provided API).
- (15%) The admin should be able to create evaluation templates. The template should contain questions which will be submitted to students to answer. Questions could either relate to the course in general, or to individual teachers in the course. For further description on what these questions could look like, see the API description.
- (5%) The admin should be able to create a new evaluation, based on a template. It should be open for a given time (i.e. there should be a start and end time).
- (15%) The student should be able to answer the evaluation.
- (10%) The admin should be able to view the results of the evaluation, i.e. for each course, view the responses from students and what grade they gave each question. There should be some graphical representation of these numbers (i.e. some kind of a chart component for instance).

Technical requirements (40%):

- (15%) Fully unit tested using Karma/Jasmine or other. Code coverage determines the grade for this part (i.e. 100% code coverage -> full grade for this part). If code coverage numbers are missing, the max number of points for this part is 10%.
- (5%) Includes at least one directive (for instance, the grading component, or a chart component)
- (5%) Uses HTML5/AngularJS validation for input fields
- (10%) Responsive, should have great usability on mobile devices. F.ex. polls should have good click areas.

- (5%) Should use a CSS precompiler and the SUIT methodology for css components.
- (10%) The application should have good usability, with minimal number of mouse clicks required to finish each task, and easy to understand workflow. Students should be able to complete the evaluation with minimal effort. Feel free to experiment with adding visual cues to the UI, such as images to indicate better what options are positive and what options are negative.

Bonus points:

- More question types and a simple form builder
- Extra points will be awarded to exceptionally elegant CSS, i.e. both visually and structurally.
- Feel free to use your imagination, what would you like to see improved in the evaluation?

API description

The following is a description of what APIs are provided, make sure you also check out the object descriptions at the bottom. Note that more detailed instructions on how to connect to the API will be published as soon as they are ready.

All users:

- **/api/v1/login - POST**
  Accepts an object with 2 parameters: user and pass. Note: you should use your own usernames (i.e. each student in the course will be able to log in), and all passwords will be 123456 (i.e. maximum NSA-proof). The admin should be able to log in using the username "admin" and the same password as the others (yes, maximum security indeed).

  If login is unsuccessful, this returns HTTP 401.
  If login is successful, this returns HTTP 200, with the following data in the response body:
    - token: an authorization token (which should be added to the HTTP headers in subsequent requests - will be explained when API is published).
    - role: a string which could be either "admin" or "student". You can use this to customize the UI after the user has logged in, i.e. show/hide menuitems accordingly.

  Note that the token returned from this API should then be added to the HTTP header of all subsequent requests (examples will be provided). If this token is not present, an HTTP 401 response will be returned from API's (again, see more detailed instructions later).

Admin:

- **/api/v1/evaluationtemplates - GET**
  Returns an array of evaluation templates. Each item in the array contains the following properties:
    - ID
    - TitleIS
    - TitleEN

- **/api/v1/evaluationtemplates/:ID - GET**
  Returns a single evaluation template (see below in the object description regarding the format)

- **/api/v1/evaluationtemplates/ - POST**
  Creates a new evaluation template, which can then be used to create an evaluation. The format of the object can be seen below, but be aware that there is no need to supply the ID of the template, since that will be allocated by the backend.

- **/api/v1/evaluations - GET**
  Returns an array of evaluations. Each object contains the following properties:
  - ID
  - TemplateTitleIS
  - TemplateTitleEN
  - StartDate - in ISO format
  - EndDate - in ISO format
  - Status - can be "new" (i.e. created but hasn't been opened) "open" (i.e. still open) or "closed".

- **/api/v1/evaluations - POST**
  Creates a new evaluation based on an evaluation template. The object sent with the request should contain the following properties:
  - TemplateID (returned when a template is created)
  - StartDate - a date in ISO format, specifies when the evaluation should open.
  - EndDate - a date in ISO format, specifies when the evaluation should close.

- **/api/v1/evaluations/:ID - GET**
  Returns a single evaluation with a given ID. The format of the response is as follows:
  - ID
  - TemplateTitleIS
  - TemplateTitleEN
  - Questions - an array where each item in the array contains the results for a given question. Each result contains the following properties:
    - ID - the ID of the question (integer)
    - TextIS
    - TextEN
    - TeacherSSN - can be empty if the question does not apply to a given teacher
    - Type (see below, can be one of the following: "text", "single", "multiple")
    - Results - if the question is a text question, the result is an array of strings:
      ["Answer from one student", "answer from another student", etc.]
      If the question is a multiple-choice question, the response contains information about how many answers each option had:
      [{ "Answer": id, "Weight": value, "Count": numberOfAnswers }]

Students

- **/api/v1/my/evaluations - GET**
  Returns an array of open evaluations. Each object contains the following properties:
  - ID
  - CourseNameIS
  - CourseNameEN
  - SemesterID

- **/api/v1/courses/:courseID/:semesterID/evaluations/:evalID - GET**
  Returns a given evaluation. The object contains the following properties:
  - ID
  - CourseQuestions - a list of questions directed towards the course in general. Could be an empty array, if no such questions should be asked.
  - TeacherQuestions - a list of questions which should be directed towards the teachers in the course. Note that the student doesn't have to evaluate all teachers, only those (s)he has interacted with.

- **/api/v1/courses/:courseID/:semesterID/evaluations/:evalID - POST**
  Saves an evaluation from a student. The request should contain an array of the answers from the student, where each item in the array contains the following properties:
  - QuestionID
  - TeacherSSN - empty if not applicable
  - Value - a text, can be a string or the ID of the option(s) selected by the student (i.e. a comma-separated list of ID's if it is possible to choose multiple answers)

- **/api/v1/courses/:courseID/:semesterID/teachers - GET**
  Returns an array of teachers in a given course. Each item in the array will contain the following properties:
  - Name
  - SSN
  - ImageURL

Object descriptions

Evaluation template:

- ID - a unique identifier (integer)
- TitleIS - a title for the template. Example: "Miðannarkennslumat".
- TitleEN - same as TitleIS, but in English
- IntroTextIS - an introduction text (in Icelandic)
- IntroTextEN - an introduction text (in English)
- CourseQuestions - an array of questions (see below). These questions should be directed towards the course in general. There may be 0 questions in this category.
- TeacherQuestions - an array of questions which will be directed towards the teachers in the course, and will be repeated for each teacher.

Question:

- ID - a unique identifier (integer)
- TextIS - the text of the question (in Icelandic)
- TextEN - the text of the question (in English)
- ImageURL - a URL to an image which should be displayed with the question (could be empty if no image is supposed to be used).
- Type - can be one of the following: "text" (for a simple text question where the user/student can reply by typing in some text), "single" (for a multiple choice question with a single answer), or "multiple" (for a multiple choice question where the student can mark multiple options)
- If Type is either "single" or "multiple", there should be an additional property called "Answers", which is an array where each item has the following properties:
  - ID
  - TextIS
  - TextEN
  - ImageURL - an image which should be displayed alongside the option
  - Weight - a number on the scale 1 to 5