UNIVERSITY OF GOTHENBURG

CHALMERS UNIVERSITY OF TECHNOLOGY

Written Examination

DIT341 – Web Development

Monday, October 25th, 2019, 08:30 - 12:30

Examiner: Philipp Leitner

Contact Persons During Exam:
Philipp Leitner (+46 733 05 69 14)
Joel Scheuner (+46 733 42 41 26)

Allowed Aides:
None except English dictionary (non-electronic), pen/pencil, ruler, and eraser.

Results:
Exam results will be made available no later than 15 working days after the exam date through Ladok.

Total Points: 100

Grade Limits: 0 – 49 points: **U**, 50 – 84 points: **G**, >=85 points: **VG**

Review:
The exam review will take place latest three weeks after the exam results have been published in Ladok. It will be announced on Canvas at least one week in advance.

1 Backend Development (18P)

Complete the code snippet of an Express app on the next page of the exam paper. Implement two endpoints:

1. One endpoint that allows to create new users. New users should be passed as JSON document in the body of the request (in the same format as the examples in the array `users`). Append the new user to the array. Return the correct status code to indicate that a new resource has successfully been created, and a JSON document of the form {`id : arrayindex` }, where ”arrayindex” shall be the position in the array where the new resource has been inserted.

2. One endpoint that allows updates of existing users. This endpoint should expect a body that contains either a name, a role, or both in the same JSON format as before. The endpoint should find the correct user in the array via the user ID given as path parameter and overwrite it with the object provided in the body of the method. The endpoint should return the correct status code for a successful operation and the newly updated resource in JSON format.

You do not need to handle any error conditions in your endpoints. Specifically, the second endpoint does not need to consider the case if the provided path parameter is not a legal array index.

Write on an extra paper all code that should be inserted into the template below to realize this behavior. Do not edit or remove existing code outside of the ”blanks” (the missing HTTP methods in lines 8 and 16, lines 9 – 13, and lines 17 – 21). Use line numbers to clarify where your code shall be inserted. Available space is not necessarily indicative of how much code is required.

```
var express = require('express');
var app = express();

var users = [
    {name : 'Philipp', role : 'Teacher'},
    {name : 'Joel', role : 'TA'}];

app.___('/canvas/users', function(req, res) {





});

app.___('/canvas/users/:user', function(req, res) {





});

app.listen(3000);
```

Figure 1: Complete the Blanks (Express)

2 REST APIs (21P)

Canvas is a learning management system used to administer courses. Imagine that Canvas provides an API for front-end development following the REST architectural style, as discussed in the course:

```
GET /canvas/users
GET /canvas/users/:user
GET /canvas/users/:user?update=name&set=Leitner
POST /canvas/courses
POST /canvas/courses/:course/assignments
```

Figure 2: Excerpt of an Imaginary Canvas API

Answer the following questions about this API, assuming that it has been designed following the REST principles introduced in the lecture.

Q2.1: Describe what functionality and behavior you would expect from the following two API methods. What is the difference between them, and how would you expect to be able to use them? **(4P)**

- POST /canvas/courses
- POST /canvas/courses/:course/assignments

Q2.2: What would a correct HTTP/1.1 request for the API method below look like? You can select an arbitrary user id. Assume in your HTTP request that your client can only handle a JSON response. **(4P)**

- GET /canvas/users/:user

Q2.3: Assume that you use the following API operation in your program. Name two status codes *not* indicating success that your application should be prepared for, and discuss briefly which erroneous situations they would represent. **(4P)**

- POST /canvas/courses/:course/assignments

Q2.4: Inspect the API method below. Do you consider it good RESTful design? How could it be improved? **(2P)**

- GET /canvas/users/:user?update=name&set=Leitner

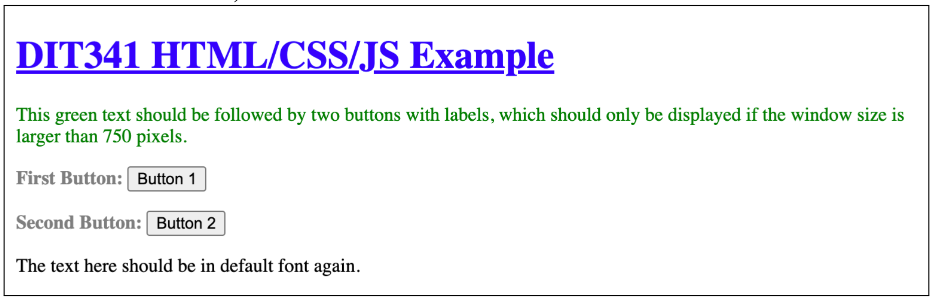
Q2.5: Define the concepts of safety and idempotency (for HTTP methods in general). Which of the 5 operations in the above API excerpt would you expect to be safe, and which would you expect to be idempotent? Provide an example of an additional method (following the same patterns as the existing methods) that would be idempotent (but not safe). **(7P)**

3 HTML, CSS, and Javascript (21P)

Complete the code snippet of an HTML / CSS / Javascript document on the next page of the exam paper. Complete the HTML page so that the following behavior is realized:

- The header should be printed in blue and underlined.
- The first paragraph of text should be printed in green.
- This should be followed by two buttons with labels. The label should be printed in bold and grey. If the buttons are pressed, the function `buttonPressed` should be called with the correct parameter (1 for the first button, 2 for the second).
- The last line of text should be printed in default font.
- The buttons and labels should only be rendered if the window size is at least 750 pixels. Otherwise this content should be hidden.

Please see the screenshot below for an example of what the page should look like for a window size with > 750 pixels (for smaller screens it's the same, but without the buttons and labels).



DIT341 HTML/CSS/JS Example

This green text should be followed by two buttons with labels, which should only be displayed if the window size is larger than 750 pixels.

First Button: Button 1

Second Button: Button 2

The text here should be in default font again.

Write on an extra paper all code that should be inserted into the template below to realize this behavior. Do not edit or remove existing code outside of the "blanks" (the missing CSS selectors in lines 3 – 5, lines 6 – 9, and lines 27 – 30). Use line numbers to clarify where your code shall be inserted. Available space is not necessarily indicative of how much code is required.

```
<html><head>
    <style>
      ____ { color: blue; text-decoration: underline }
      ____ { color: green; }
      ____ { color: grey; font-weight: bold; }



    </style>
    <script>
        function buttonPressed(buttonNr) {
          console.log('Button '+buttonNr+' pressed.');
        }
    </script>
  </head>
  <body>
    <h1> DIT341 HTML/CSS/JS Example </h1>

    <div green>
      <p>
        This green text should be followed
        by two buttons with labels,
        which should only be displayed if the
        window size is larger than 750 pixels.
      </p>
    </div>




    <p> The text here should be in default font again. </p>

  </body>
</html>
```

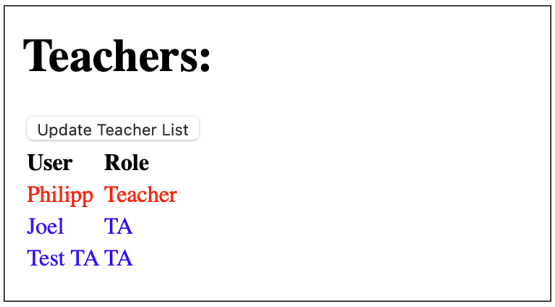
Figure 3: Complete the Blanks (HTML/CSS/JS)

4 Frontend Development (12)

Complete the code snippet of a Vue.js app on the next page of the exam paper. Implement the following behavior:

- When pressing the button, the method ”updateTeacherList” shall be called.
- The table shall be completed to print users and their roles, as specified in the ”users” array (one teacher per row).
- Rows for users with the role ”Teacher” shall be printed in red, rows for TAs shall be printed in blue (use the predefined CSS styles). *Tip:* you can use the `v-bind` directive to dynamically bind attributes in Vue.js, with the syntax `v-bind:<attribute>="<JS code>"`.

Refer to the screenshot below for what the finally generated site shall look like.



Teachers:

Update Teacher List

User	Role
Philipp	Teacher
Joel	TA
Test TA	TA

Write on an extra paper all code that should be inserted into the template below to realize this behavior. Do not edit or remove existing code outside of the ”blanks” (the missing code on lines 9 and 11, as well as the implementation of the table in lines 15 to 17). Use line numbers to clarify where your code shall be inserted. Available space is not necessarily indicative of how much code is required.

```
<html>
<!-- assume vue.js is imported -->
<style>
  .Teacher{ color: red; }
  .TA{ color: blue; }
</style>
<body>

  <div ___>
    <h1>Teachers:</h1>
    <button type="button" @click="___">Update Teacher List</button>

    <table>
      <tr style="font-weight: bold;"><td>User</td><td>Role</td></tr>



    </table>
  </div>
  <script>
        var app = new Vue({
            el: '#vueapp',
            data: {
              users: [ {name : 'Philipp', role : 'Teacher'},
                        {name : 'Joel', role : 'TA'},
                        {name : 'Test TA', role : 'TA'} ]
            },
            methods: {
                updateTeacherList: function () {
            // assume that this updates 'users'
                }
            }
        });
    </script></body></html>
```

Figure 4: Complete the Blanks (Vue.js)

5 REST (8P)

Name and briefly describe four of the important architectural constraints that define the REST architectural style.

6 Single Page Apps (6P)

Imagine you are building a Canvas-like Web-based system. Briefly describe and compare how this system could be implemented as a ”traditional” Web application or as a Single Page Application (SPA). Particularly focus on the difference in responsibilities between client and server in both styles, as well as on the communication between client and server.

7 Cookies (8P)

Imagine you are building a Canvas-like Web-based system, with an API similar to the one discussed in Question 2. What could you use cookies for in such a system? Describe how cookies work in general, what functionality in your Canvas clone could be implemented using cookies, and provide at least one example HTTP interaction using the endpoints from Question 2 to illustrate how cookies work on HTTP protocol level.

8 Networking Protocols (6P)

Name and briefly describe the two important OSI layer 4 and 5 protocols we discussed when discussing the OSI layered network stack. Particularly describe how these two protocols differ, and name at least one common use case for each of the protocols.