UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Written Examination

DIT341 – Web and Mobile Development

Tuesday, January 5th, 2021, 14:00 - 18:30

Examiner: Philipp Leitner

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

Allowed Aides:
All material can be used in this home exam. However, the exam is individual, and collaborating or sharing solutions with other students is strictly forbidden.

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 over Zoom following publishing of grades.

1 Backend Development (20P)

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

1. An endpoint that returns all elements in the array `json_array` in JSON format.
2. An endpoint that accepts an ID as path parameter and returns the element of `json_array` that has an ID property equal to the passed parameter. If the array does not contain an element with this ID the correct status code along with the error object (in JSON format) `{'Error': 'Invalid Id X'}` (where X is the passed parameter value) should be returned.
3. And an endpoint that accepts a JSON object as body. If the passed JSON object does not have a property `id` the endpoint shall reject the value and return the status code 400 along with the error object `{'Error':'No id'}` (again in JSON format). If the passed object has an `id` the object shall be appended to the array, and the correct status code along with the passed object shall be returned.

Use suitable HTTP methods to implement these endpoints. You do not need to handle error cases aside from the ones explicitly mentioned above.

> Either submit the complete program, or write down only the missing code (in this case refer to the lines in the template via line numbers). You are not allowed to modify existing code or add code outside of lines 11 – 28 in the template. Available space is not necessarily indicative of how much code is required. The code snippet is also available on Canvas as code file.

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(bodyParser.json());

var json_array = [
    {'id': 1, 'value': 'a'},
    {'id': 2, 'value': 'b'},
    {'id': 10, 'value': 200}
];









function getElementById(id) {
    return json_array.find(function(element) {
        return element.id == id;
    });
}

app.listen(3000);
```

Figure 1: Complete the Blanks (Express)

2 APIs (17P)

GitHub is a platform for managing source code. Find below a (strongly simplified) excerpt from the GitHub API.

```
GET /users/
GET /:organization/repositories
GET /:organization/repositories/:repo
POST /:organization/repositories/
PATCH /:organization/repositories/:repo
PUT /:organization/repositories/:repo
```

Figure 2: Excerpt of a GitHub-inspired 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 the functionality, behavior, and potential status codes you would expect from the following two API methods. **(4P)**

- GET /:organization/repositories
- GET /:organization/repositories/:repo

Q2.2: What would be a good REST endpoint pattern (following the existing examples) to use to remove a specific user? Provide the pattern using the same notation as above, and explain briefly. **(2P)**

Q2.3: How would (when using one of the GET methods) a user specify what representation the server should return? Enumerate two representations we have frequently used or discussed in class. **(4P)**

Q2.4: Write down a legal, successful HTTP request for the following endpoint (the

complete HTTP request message as it will be transferred on the network). You expect an HTML response, and the server has previously set the cookie `session=cA111`. **(4P)**

- GET /:organization/repositories/

Q2.5: Imagine that this system uses GraphQL as an alternative to this REST API. Briefly describe GraphQL, and discuss what concrete advantages and challenges would arise from using GraphQL in this context. **(3P)**

3 CSS (14P)

Write the CSS style definitions in the HTML page according to the following rules:

1. `p` elements should always contain blue text (use `text-color: blue`),
2. except for `p` elements that are directly contained in an `div` element, which should use red instead.
3. Elements with the id "yellowtext" shall contain yellow text.
4. Elements with the class "greentext" shall contain green text.
5. `p` elements with an attribute `bordered` should have a framed box around them (use `border: solid`).
6. Elements with the class "hideable" shall be hidden if the screen size is less than or equal to 1000px.

Insert the required CSS code within the `style` definition, and refer to the example elements within the body to see if they behave correctly.

Either submit the complete program, or write down only the missing code (in this case refer to the lines in the template via line numbers). You are not allowed to modify existing code or add code outside of lines 5 – 10 (in other words, all your code changes should go between the `style` tags). Available space is not necessarily indicative of how much code is required. The code snippet is also available on Canvas as code file.

```
1   <!doctype html>
2   <html>
3   <head>
4       <style>
5
6
7
8
9
10
11      </style>
12  </head>
13
14  <body>
15    <h1> An example of a CSS styled app </h1>
16
17    <p> Some text that should be blue. </p>
18
19    <div> <p> This, on the other hand, should be red.</p>
          </div>
20
21    <div id="yellowtext"> And here some yellow text. </div>
22
23    <div class="greentext"> And now: green. </div>
24
25    <p bordered>
26        Hopefully there is a frame around this blue text.
27    </p>
28
29    <p class="hideable">
30        And finally some blue text that should disappear if
              the screen is smaller than 1000px.
31    </p>
32  </body>
33  </html>
```

Figure 3: Complete the Blanks (CSS)

4 Frontend Development (16P)

Complete the code snippet of a Vue.js app on the next page of the exam paper. Complete the code so that (upon loading) the following list is dynamically rendered based on the data model of the Vue.js app:

Clicking the button "Add Course: X" should append the value from the text box on the left to the list. The label of the button should be generated dynamically based on the current value of the text box (i.e., if the user inputs "DAT335" in the input box the button label shall read "Add Course: DAT335"). Clicking the button "Remove Last Course" shall remove the last item from the list.
Your application does not need to handle any errors. The buttons and input field shall be simple HTML input elements, do not use Bootstrap or BootstrapVue in this exercise.

> Either submit the complete program, or write down only the missing code (in this case refer to the lines in the template via line numbers). You are not allowed to modify existing code or add code outside of lines 9 – 11, 13 – 15, and 26 – 31 in the template. Available space is not necessarily indicative of how much code is required. The code snippet is also available on Canvas as code file.

```
1   <!doctype html>
2   <html> <!-- assume Vue.js is imported in head -->
3
4   <body>
5       <div id="vueapp">
6
7           <h1>List of Courses:</h1>
8           <ul>
9
10
11
12          </ul>
13
14
15
16      </div>
17
18      <script>
19          var app = new Vue({
20              el: '#vueapp',
21              data: {
22                  courses: ["DIT032", "DIT341"],
23                  to_add: "New Course"
24              },
25              methods: {
26
27
28
29
30
31
32              }
33          });
34      </script>
35
36  </body></html>
```

Figure 4: Complete the Blanks (Vue.js)

5 Testing (9P)

You are building a simple clone of the GitHub system described in Q2. Describe three ways how you can test functional correctness of your system, and categorize them in the dimensions test granularity and degree of automation. Further describe how you could additionally test the performance of the system, and name and describe two concrete performance metrics that may be important to measure for this system.

6 Single Page Apps (8P)

Again considering this GitHub system, describe how this system could be implemented as a Single Page Application, and compare it to an alternative, more traditional web-based architecture. Particularly describe how browser, backend, and frontend interact, and discuss advantages and disadvantages of both architectures.

7 Networking (7P)

Name and briefly describe the seven layers of the OSI stack. Again considering the GitHub system, name a concrete protocol that would be used in the system for each of the OSI layers.

8 HATEOAS (5P)

Using the entry point provided below, discuss how the GitHub system could implement the HATEOAS principle.

- https://www.my-gh-clone.eu

9 IaC (4P)

In what ways would adopting Infrastructure-as-Code (IaC) help you when building the GitHub system? Provide specific examples of concrete benefits.