UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Written Examination

DIT341 – Web and Mobile Development

Monday, October 26th, 2020, 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:
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 (16P)

Complete the code snippet of an Express app on the next page of the exam paper. Implement a collection endpoint that allows clients to retrieve *all* elements of the array `storage`. The endpoint should support a query parameter `sort` indicating how the response shall be sorted: if a value is given for this parameter (either `asc` or `desc`) the results shall be sorted appropriately; if `sort` is omitted, the result shall be returned in the same order as stored in the array. Return a JSON object. If an invalid value is passed as sorting parameter, return the status code 400 with an appropriate error message in plain text. Select an appropriate endpoint type (HTTP method) for this kind of operation.

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). 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 app = express();

var storage = [7, 5, 8, 9, 13, 4];

app.           ('/numbers', function(req, res) {
















});

function sort_asc(array) {
    return array.sort((a, b) => a - b);
}

function sort_desc(array) {
    return array.sort((a, b) => b - a);
}

app.listen(3000);
```

Figure 1: Complete the Blanks (Express)

2 HTTP and REST APIs (22P)

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

```
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 what functionality and behavior you would expect from the following API method. What response and status code would you expect from the service in case of a successful request? What would you expect to happen if you invoked the same API method multiple times, with the same URL and body? **(4P)**

- POST /:organization/repositories/

Q2.2: 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)**

- GET /:organization/repositories/:repo

Q2.3: What would be a good REST endpoint pattern (following the existing examples) to use if a user wanted to remove a repository? Provide the pattern using the same notation as above, and explain briefly. Further, give at least *one* example of a ”technically correct” way to implement a deletion that does *not* follow good REST practices (and describe why it does not follow good practice). **(4P)**

Q2.4: 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? Why? **(4P)**

Q2.5: 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 a JSON response, and the server has previously set the cookie `session=aF36CC`. Send a simple JSON object with two fields (your choice) as body. **(6P)**

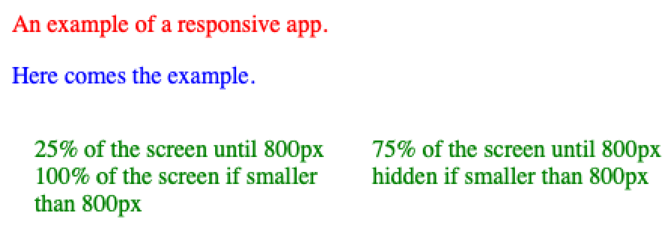
- POST /:organization/repositories/

3 Responsive Web Design and CSS (15P)

Complete the code snippet of a simple RWD demonstration on the next page of the exam paper. Add the appropriate CSS and HTML code so that the described behavior is realized, and refer to the screenshots below for expected formatting and behavior. You shall not use Bootstrap or BootstrapVue in this example.

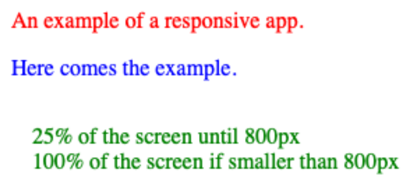
Either submit the complete program, or write down only the missing code (in this case refer to the lines in the template via line numbers). Available space is not necessarily indicative of how much code is required. The code snippet is also available on Canvas as code file.

This is the expected behavior of the HTML page when the screen size exceeds 800 pixels:



```
An example of a responsive app.

Here comes the example.

25% of the screen until 800px      75% of the screen until 800px
100% of the screen if smaller      hidden if smaller than 800px
than 800px
```

This, on the other hand, is what should be shown for smaller screen sizes:



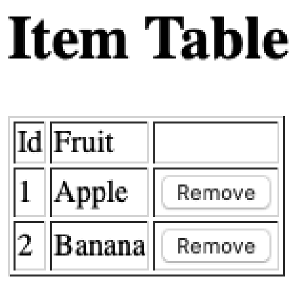
```
An example of a responsive app.

Here comes the example.

25% of the screen until 800px
100% of the screen if smaller than 800px
```

```html
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        * { box-sizing: border-box; }
        .col-3 { width: 25%; }
        .col-9 { width: 75%; }
        .col-12 { width: 100%; }
        [class*="col-"] {
            float: left;
            padding: 15px;
        }
        p { color: red; }
        #sometext { color: blue; }
        .instruction { color: green; }




    </style>
</head>
<body>
        <p         > An example of a responsive app.</p>
        <p         > Here comes the example.</p>

        <div id="left"          >
            25% of the screen until 800px <br>
            100% of the screen if smaller than 800px
        </div>

        <div id="right"         >
            75% of the screen until 800px <br>
            hidden if smaller than 800px
        </div>
</body></html>
```

Figure 3: Complete the Blanks (CSS)

4 Frontend Development (14P)

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 table is dynamically rendered based on the data model of the Vue.js app:



Item Table:

Id	Fruit	
1	Apple	Remove
2	Banana	Remove

Clicking any of the ”Remove” buttons should (a) remove this row from the table and (b) send the correct request to trigger removal to an API endpoint using the Axios library. Use the content of the variable `endpoint` as the stem of the endpoint that you invoke. Your application does not need to handle errors, nor do you need to react to the response of the server. The button can be a simple HTML button, do not use Bootstrap or BootstrapVue.

> 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). Available space is not necessarily indicative of how much code is required. The code snippet is also available on Canvas as code file.

```
<html> <head>
    <!-- assume Axios and Vue.js are loaded -->
</head>
<body>
    <div id="vueapp">
        <h1>Item Table:</h1>
        <table border="1">



        </table>
    </div>
    <script>
      var endpoint = 'http://api.mystore/fruits/';

      function removeItemFromArray(arr, id) {
        for (var i = 0; i < arr.length; i++) {
            if (arr[i].id === id) {
                arr.splice(i, 1);
            }
        }
      }

      var app = new Vue({
          el: '#vueapp', data: {
              items: [ {'id' : 1, 'name' : 'Apple'},
                         {'id' : 2, 'name': 'Banana' } ]
          },




      });
    </script>
</body> </html>
```

Figure 4: Complete the Blanks (Vue.js)

5 Sessions and Authentication (7P)

You are building a simple clone of the GitHub system described above. One of your key requirements is that users need to be able to log in (via username/password), and many of your API operations should only be available to users that have previously logged in.
Sketch briefly (e.g., using a sequence diagram) how you can use a combination of cookies and sessions to implement this authentication scheme. Also discuss how you would check (in the implementation of an endpoint) whether a user is authenticated, and what the expected behavior of the endpoint would be if the user isn't authenticated.

6 Accessibility (6P)

Discuss three concrete accessibility challenges in your GitHub clone that a user with a broken right arm (assume the user is wearing a rigid cast, and that the right hand is their dominant hand) may face. How could your application support this user?

7 Domain Name Service (8P)

Discuss the key elements of the following DNS name. Further, describe the procedure how the operating system will resolve this DNS name to an IP address (ignoring cached values). What are the advantages of this way of handling DNS resolution?

- www.my-gh-clone.eu

8 GraphQL (6P)

You are considering building a GraphQL API (rather than a REST API as discussed in Question 2) for your GitHub clone. Discuss possible advantages and disadvantages. Imagine a scenario where choosing GraphQL may be the better option.

9 Cloud Computing (6P)

Discuss, based on the example of the GitHub clone, the concepts of scaling up/down as well as out/in. What problem do these ideas solve, and what are their limitations?