UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Written Examination

DIT342 – Web Development

Wednesday, January 3, 2024, 14:00 - 18:00

Examiner: Philipp Leitner

Contact Persons During Exam:
Magnus Ågren (+46 733 35 22 92)
Backup:
Philipp Leitner (+46 733 05 69 14)

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 – 69 points: **3**, 70 – 89 points: **4**, $\geq$ 90 points: **5**

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)

The code snippet (Figure 1) on the next page of the exam paper shows parts of a simple shopping list Express app (see also Figure 2 for inspiration). Complete the implementation of the two endpoints:

1. One endpoint for fetching existing shopping lists. Return the correct element from the array `shoppingLists`, in JSON format, with the appropriate status code. The shopping list id is the array index. If the list cannot be found, send an error response. Note the availability of error handling middleware.

2. One endpoint that allows appending items to an existing shopping list. Items are passed in the body of the request, as a JSON document of the form
   `{ item: "new item" }`
   Append the new item to the existing ones in the particular shopping list. Return the correct status code to indicate that a new resource has successfully been created, and a JSON document with all the items in the particular shopping list. As above, if the list cannot be found, send an error response. Again, note the availability of error handling middleware.

Write on an answer sheet of 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 11 and 18, lines 12 – 15, and lines 19 – 22). 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 bodyParser = require('body-parser');
var app = express();
app.use(bodyParser.json());

var shoppingLists = [
  { title: 'Food', items: ['Turnips', 'Cookies', 'Jam'] },
  { title: 'Furniture', items: ['Comfy chair', 'Bar'] }
];

app.___(___, function(req, res, next) {




});

app.___(___, function(req, res, next) {




});

// Common error handler
app.use(function (req, res) {
    res.status(404).json({ 'message': 'Not Found' });
});

app.listen(3000);
```

Figure 1: Complete the Blanks (Express)

2 REST APIs (20P)

Consider the API for a shopping list service, see Figure 2. Assume it adheres to the REST architectural style, and could be used from multiple front-ends:

```
GET /shoppinglists/:id
POST /shoppinglists
POST /shoppinglists/:id/items
PUT /shoppinglists/:id/items/:item
DELETE /shoppinglists/:id/items/:item
```

Figure 2: Example Shopping List API Excerpt

Answer the following questions about this API:

Q 2.1: (5P) How would you expect to be able to use the following endpoints – what would you expect to send and receive? What is the difference between the two?

```
POST /shoppinglists/:id/items
PUT /shoppinglists/:id/items/:item
```

Q 2.2: (2P) Consider the following HTTP request to the API.

```
POST /shoppinglists/:id
```

What response would you expect, and why – what potential problems would this request have in practice?

Q 2.3: (4P) Consider the following HTTP request to the API.

```
GET /shoppinglists/:id
```

Describe the differences between a response with status code 404 and a response with status code 503. What possible fixes are there from the client side?

Q 2.4: (9P) For HTTP methods in general, define what it means for a method to be safe, idempotent, or unsafe. For the 5 operations in Figure 2, which of these terms apply to each? Provide an example of an additional method (following the same patterns as the existing methods) that would be idempotent (but not safe).

3 Web Application Kinds (8P)

Contrast Single Page Applications (SPA) with the more traditional way of building web applications (from the early days of the Web). Describe the two approaches and highlight their key differences.

4 Responsive Web Design (9P)

Describe the terms *Responsive Web Design (RWD)*, *Web Accessibility*, and *Reactivity*. Contrast these by giving one example per term, unique to that term. (Hence, for RWD, something that is RWD but not Web Accessibility nor Reactivity, and so on for each term).

5 Understanding CSS Formatting (12P)

Consider the HTML document in Figure 3. Draw the resulting layout for the two cases:

1. when the page is shown on a narrow mobile device.
2. when the page is shown on a wide computer monitor (using the entire screen).

Describe also, per case, in which colors the browser will render the text. Assume that the default browser color is black. You can refer to line numbers when providing your answers.

```html
!DOCTYPE html>
<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%; }
        div { float: left;
              border:solid; }
        #left { color: pink; }

        @media screen and (max-width: 800px) {
            [class*="col-"] {
                width: 100%;
                color: blue;
            }
            #right { color: orange ;}
            #next { display: none ;}
        }
    </style>
</head>
<body>
    <div id="left" class="col-9">
        An exciting paragraph
    </div>
    <div id="right" class="col-3">
        Some more writing
    </div>
    <p id="next"> Next line</p>
</body>
</html>
```

Figure 3: HTML and CSS

6 Frontend Development (8P)

Complete the code snippet of a Vue.js app shown in Figure 4 as follows. When pressing the button, the `fetch` method shall be called. Below the heading, render the contents of `items` in an unordered list. Place a button next to each item. When pressed, this button shall invoke the `remove` function to remove that particular item from the list.

Write on an answer sheet of 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 line 6, and lines lines 10 – 13). 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 and Axios are imported in Head -->
<body>
  <div id="vueapp">
    <p>
      <button _____=_____>Fetch shopping list</button>
    </p>
    <p>
      <h1>Shopping list</h1>




    </p>
  </div>

  <script>
    var app = new Vue({
      el: '#vueapp',
      data: {
        items: [],
      },
      methods: {
        fetch: function() {
          // assume this function populates this.items
        },
        remove: function(item) {
          // assume this function removes the given item
          // from this.items
        }
      }});
  </script>
</body>
</html>
```

Figure 4: Complete the Blanks (Vue.js)

7 Internet (7P)

The OSI model gives a layered architecture for network communication, with each layer providing further abstraction on top of the layers below. In this context, consider the protocols HTTP, TCP, and IP; what does each protocol do? Your answer doesn't need to describe specific layers of the OSI stack, just the general functionality or abstraction provided by each protocol.

8 Cloud Computing (10P)

How does cloud deployment relate to the REST constraint of statelessness? Your answer should cover the following terms: Elasticity, Load balancing, Infrastructure-as-Code, and Infrastructure-as-a-Service.

9 Testing Strategies (8P)

Imagine you are providing a web service for travel planning. This service consists of a web app, using a REST API which you are also providing direct access to. The API handles, for example, departure and arrival information, and route planning for combinations of traffic types (railroad, buses, air, etc.). Assume that both your app and your API already have established user bases. A new feature, providing information on delays, is to be added to your service.
Describe how this new feature could be tested before deployment. Cover four separate aspects that together test both the app and the API. *Tip:* If you find it helpful, draw a picture of how the different test approaches map to the different parts of your service.