UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Written Examination

DIT342 – Web Development

Wednesday, January 8, 2025, 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**

1 Backend Development (27P)

The code snippet (Figure 1) on the next page shows parts of a simple Express app for booking car rentals. See also Figure 2 for inspiration. Implement two endpoints:

1. One endpoint that allows creating new bookings. Booking details are passed in the request body as a JSON document of the form
   ```
   { car:   licence plate number,
     from:  booking start date,
     to:    booking end date  }
   ```
   Use the provided function `isAvailable` to check if the sought car is bookable during the given dates. If the car is available, add a new booking to the list of bookings. Return the status code indicating successful creation of a new resource, and the new booking as a JSON document, including the length of the `bookings` array as id. If the car is not available, return an appropriate error code and an error message. You only need to handle this particular error case.

2. One endpoint that allows editing existing bookings. The booking id, sent as a path parameter, is an index of the `bookings` array. The request body contains a JSON document with updated booking details. Any combination of car, from date, and to date can be sent. If the booking is found, overwrite the existing properties with those received. Assume the update is possible; you don't need to check the availability for the updated information. Return the status code indicating a successful operation, and the newly updated booking in JSON format. If the booking cannot be found, return an appropriate error code and an error message. You only need to handle this particular error case.

You only need to implement error handling for the specific conditions described above.

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 20, lines 12 – 17, and lines 21 – 26). Use line numbers to clarify where your code shall be inserted. Available space is *not* necessarily indicative of how much code is required.

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

var bookings = [
 {'car': 'abc123', 'from': '2025-01-22', 'to': '2025-01-23'},
 {'car': 'foo798', 'from': '2025-01-30', 'to': '2025-02-03'}
];

app.______(_______________, function(req, res) {






});

app.______(________________, function(req, res) {






});

/* This function checks if the car is available between the
   provided dates. Returns a boolean. */
function isAvailable(car, from, to);

app.listen(3000);
```

Figure 1: Complete the blanks (Express)

2 REST API Example (6P)

Figure 2 shows parts of a simplified API for a booking app for cars. Assume it adheres to the REST architectural style.

```
POST /bookings
GET /bookings
GET /bookings/:id
DELETE /bookings
DELETE /bookings/:id
```

Figure 2: Booking App API Excerpt

Q 2.1: (3P) Consider the following HTTP request to the API.

```
DELETE /bookings/75
```

What do the response codes 401, 404, and 500, respectively, indicate in this case?

Q 2.2: (3P) How could an endpoint in Figure 2 be extended with query parameters to allow checking availability between two dates *from* and *to*?

3 RESTful Architecture (13P)

Q 3.1: (3P) Describe the *Statelessness* constraint of REST.

Q 3.2: (8P) The *Uniform Interface* constraint consists of four sub-constraints

1. Identification of resources
2. Manipulation of resources through representations
3. Self-descriptive messages
4. Hypermedia as the engine of application state (HATEOAS)

Describe each of these.

Q 3.3: (2P) Explain what it means if an HTTP method is expected to be unsafe.

4 CSS Formatting (11P)

Consider the HTML document in Figure 3. Describe 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.

```
 1  <!doctype html>
 2  <html>
 3    <head><style>
 4      div { color: orange; }
 5      div > span { color: green; }
 6      p[custom="pink"] { color: pink; }
 7      #highlight { color: red; }
 8      .contrast { color: blue; }
 9    </style></head>
10  <body>
11    <div>This line starts the body text.</div>
12    <div class="contrast">The second line contrasts,</div>
13    <p>Next paragraph
14      <span>with specific markings.</span>
15    </p>
16    <p custom="pink">Standout paragraph
17      <span id="highlight">with highlights.</span>
18    </p>
19    <div class="contrast">
20      <div>Further subdivided text</div>
21      <div>Next subdivision
22        <span>with marked text.</span>
23      </div> After the subdivisions.
24      <span style="color: yellow">also marked text.</span>
25    </div>
26  </body></html>
```

Figure 3: HTML and CSS

5 Frontend Development (25P)

Complete the code snippet of a simple inventory management Vue.js app in Figure 5. Implement the following behavior:

- When the *Add item* button is clicked, the `addToInventory` function shall be called. This function shall add the new item and its starting amount to the inventory. For simplicity, assume that the provided values are correct; you don't need to implement any error handling.
- Render the items and their amounts from the inventory an unordered list. Place a plus- and a minus-button next to each item. Clicking the plus-button shall invoke the `increase` function, which shall increase the amount of that item in the inventory by one.
- Clicking the minus-button shall invoke the `decrease` function, which shall decrease the amount of that item in the inventory by one, if the amount is greater than zero. If the amount is zero, it should not be decreased further.

Refer to the screenshot in Figure 4 for what the generated site shall look like. *Hint:* the `v-for` directive can get both the current element and its index when iterating over an array, using the syntax `v-for="(element, index) in array"`.

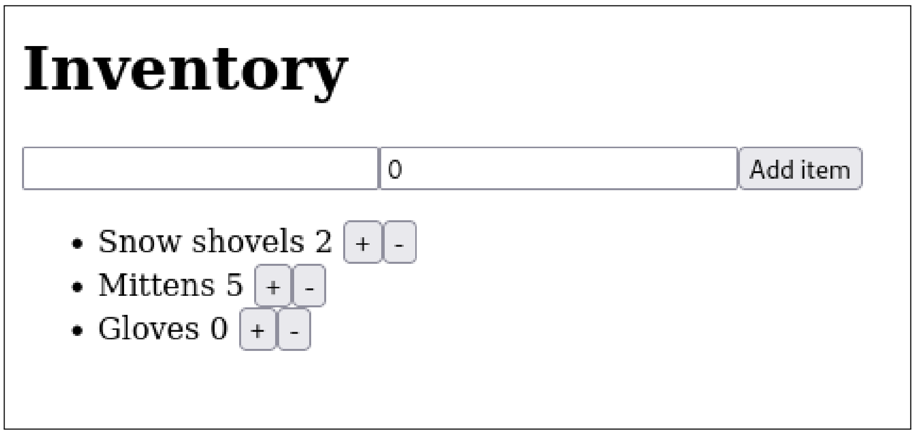

Figure 4: Vue.js example

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 6, 7, 8, lines 11 – 14, and the implementation of the functions `addToInventory`, `increase`, and `decrease`), lines 26 – 34. 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><!-- assume vue.js is imported --></head>
<body>
  <div id="vueapp">
    <h1>Inventory</h1>
    <p>
      <input type="text" _________=_________ />
      <input type="text" _________=_________ />
      <button @click="_________">Add item</button>
    </p>
    <p><ul>
      <li __________________>



      </li>
    </ul></p>
  </div>
  <script>
    Vue.createApp({
      data() { return {
        inventory: [],
        newItem: "",
        startAmount: 0,
      }},
      methods: {
        addToInventory: function() {

        },
        increase: function(_________) {

        },
        decrease: function(_________) {

        }
      }}).mount('#vueapp')
  </script></body></html>
```

Figure 5: Complete the blanks (Vue.js)

6 Communication Protocols (9P)

Describe the difference between the following communication protocol properties:

- multicast vs. unicast
- ordered vs. unordered
- reliable vs. unreliable

Which of these properties does UDP have?

7 Frontend Technology (9P)

Answer the following questions about frontend technology.

Q 7.1: (2P) Describe the notion of *Reactivity* as found in Vue.js, from your developer point-of-view implementing a webapp with Vue.js.

Q 7.2: (2P) How does Vue achieve Reactivity? *Hint:* consider the Document-Object-Model.

Q 7.3: (2P) Describe the notion of *Responsive Web Design*.

Q 7.4: (3P) How can Responsive Web Design be implemented in practice, for example with Bootstrap?