UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Written Examination

DIT342 – Web Development

Wednesday, August 20, 2025, 08:30 - 12:30

Examiner: Philipp Leitner

Contact Persons During Exam:
Raffaela Groner (+46 70 309 71 70)
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 (19P)

The code snippet (Figure 1) on the next page shows part of a public transport information app (see Figure 2 for inspiration). Implement two endpoints:

1. One endpoint that allows editing of existing records of public transport routes. Record updates are passed in the body of the request as a JSON document containing either a new description of the route's name, an updated list of stops, or both. By matching the id with the id of the route given as a path parameter, the endpoint should find the correct record to update in the array of `routes` and overwrite existing properties with those received in the request body. Return the correct status code for a successful operation, and the newly updated route in JSON format.

2. One endpoint that allows appending new stops to an existing record of a route. Stops are passed in the body of the request as a JSON document of the form: `{stops: new stop}`
   Append the new stop to the existing ones of the route's record. Return the correct status code to indicate that a new resource has successfully been created, and the newly updated route in JSON format.

You do not need to implement any error handling.

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 12 and 24, lines 13 – 20, and lines 24 – 31). 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 bodyParser = require('body-parser');
app.use(bodyParser.json());
var routes = [
  {"id": "64026", "name": "RÖD", "lastUpdate": "2025-04-07",
      "stops": [{"name":"Lilla Varholmen", "platform": "A"},
         {"name": "Majvik", "platform":"B"}]},
  {"id": "86475", "name": "16", "lastUpdate": "2025-03-19",
      "stops": [{"name":"Fyrktorget", "platform":"C"},
         {"name":"Tolvskillingsgatan", "platform":"A"}]}];

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








});

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








});

app.listen(3000);
```

Figure 1: Complete the blanks (Express)

2 REST APIs (18P)

Figure 2 shows parts of a simplified API for a public transport information app. Assume it adheres to the REST architectural style. You can also assume that ids of routes and the names of stops function as unique identifiers.

```
POST /routes
PUT /routes
```

Figure 2: Public transport information API excerpt

Q 2.1: (2P) What functionality and behavior would you expect from the following two API methods?

```
POST /routes
PUT /routes
```

Q 2.2: (4P) Various conventions must be adhered to when designing a RESTful API. A number of these are violated in the following definition. Specify the correct version of the RESTful API.

```
GET /route?delete(:route)
GET /route/:route/getAllStops
```

Q 2.3: (2P) Describe briefly what the REST-Constraint Stateless means.

Q 2.4: (3P) Assume that the API in Figure 2 is extended to allow filtering bus stops based on the platform they stop at. What request do you have to send to get all stops that are on the route `64026` and stop at platform *B*?

Q 2.5: (4P) What does the acronym CRUD stand for? Name for each CRUD function one HTTP method that realizes CRUD behavior.

3 CSS Formatting (10P)

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>
 4    <style>
 5      div, a { color: purple; }
 6      div a { color: pink; }
 7      .new { color: red; }
 8      #new { color: orange; }
 9      div div { color: blue; }
10      a.link.schedule { color: green; }
11    </style>
12  </head>
13  <body>
14    <h1>Trafik Information</h1>
15    <div>
16      <span id="new"> New bus stops </span>
17      will be added to the
18      <a class="link" href="lineInfo.html">line 16.</a>
19      <h1 style="color: yellow;">New Schedule</h1>
20      The new
21      <a class="schedule" href="schedule.html">timetable is</a>
22      <span class="new">now available</span>.
23      <div class="link schedule">to construction work, the
          departure times of</div>
24      <a class="link schedule" href="lineInfo.html">line 1 have
           changed.</a>
25    </div>
26  </body>
27  </html>
```

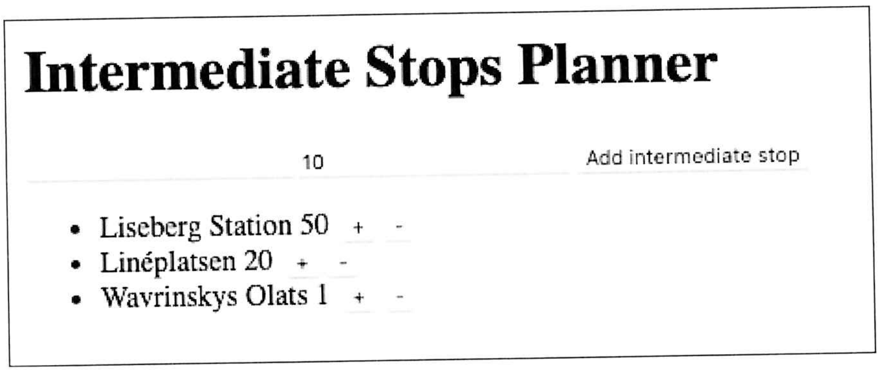
Figure 3: HTML and CSS

4 Frontend Development (25P)

Figure 5 shows a simplified Vue.js app for planning several intermediate stops on a bus trip. Implement the following behavior:

- When the *Add intermediate stop* button is clicked, the `addToStops` function shall be called. This function shall add the new intermediate stop and the desired duration of stay to the list of stops. For simplicity, assume that the provided values are correct; you don't need to implement any error handling.
- Render the intermediate stops and their duration of stay from the list of stops as an unordered list. Place a plus- and a minus-button next to each intermediate stop. Clicking the plus-button shall invoke the `increase` function, which shall increase the duration of stay of that stop in the list of stops by one.
- Clicking the minus-button shall invoke the `decrease` function, which shall decrease the duration of stay of that stop in the list of stops by one, if the duration is greater than one. If the duration is one, 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"`.



Intermediate Stops Planner

10 Add intermediate stop

- Liseberg Station 50 + -
- Linéplatsen 20 + -
- Wavrinskys Olats 1 + -

Figure 4: Vue.js example

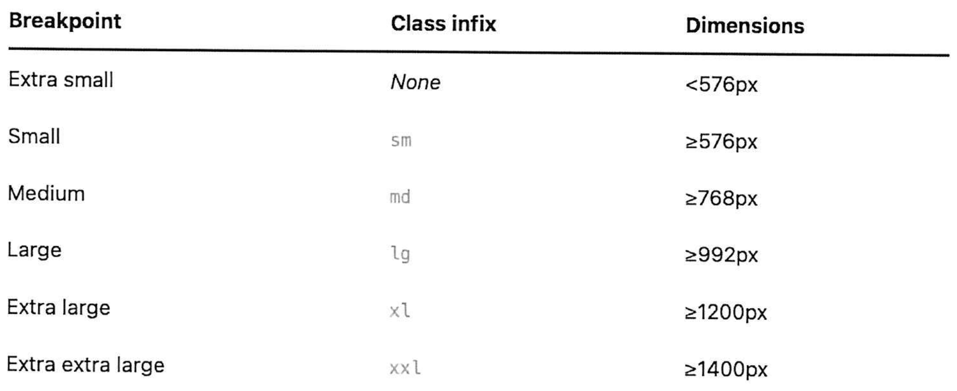
Write on an answer sheet of paper all code that should be inserted into the template below to realize this behavior. Do not edit or remove existing code outside of the "blanks" (the missing code on lines 6, 7, 8, lines 11 – 14, and the implementation of the functions `addToStops`, `increase`, and `decrease`), 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>Intermediate Stops Planner</h1>
    <p>
      <input type="text" __________=__________ />
      <input type="text" __________=__________ />
      <button @click="__________">Add intermediate stop</button>
    </p>
    <p><ul>
      <li ____________________>



      </li>
    </ul></p>
  </div>
  <script>
    Vue.createApp({
      data() { return {
        stops: [],
        newStop: "",
        newDuration: 10,
      }},
      methods: {
        addToStops: function() {

        },
        increase: function(__________) {

        },
        decrease: function(__________) {

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

Figure 5: Complete the blanks (Vue.js)

5 Responsive Web Design (10P)

Consider the code snippet in Figure 9 where a website layout is defined using the Bootstrap grid system. Figure 7 sketches a Bootstrap grid layout for a browser window larger than 1300px. Figure 8 sketches a Bootstrap grid layout for a browser window on a mobile device with 530px. In both figures, the div element in line 15 in Figure 9 is already sketched in correctly. Note that the thicker border highlights the width of the div-element in the grid.
On an answer sheet of paper, draw a copy of the two Bootstrap grid layouts, including the div-element already drawn, and add the remaining div-elements in each drawing. Make sure that the position and width of the respective divs in the grid are clearly recognizable, and annotate which sketch should represent which browser size. Only paint the div elements affected by the CSS code in lines 7 to 9 in Figure 9. Figure 6 provides an overview of the different breakpoints in Bootstrap.



Breakpoint	**Class infix**	**Dimensions**
Extra small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

Figure 6: Bootstrap breakpoints

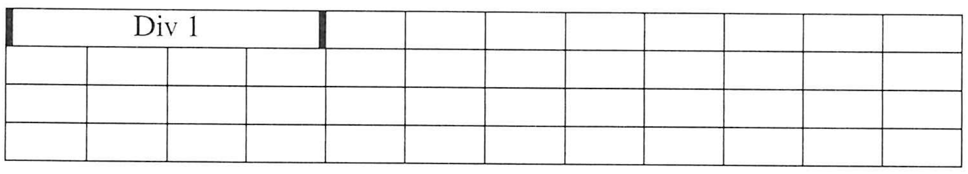


Figure 7: Large browser window – 1300px

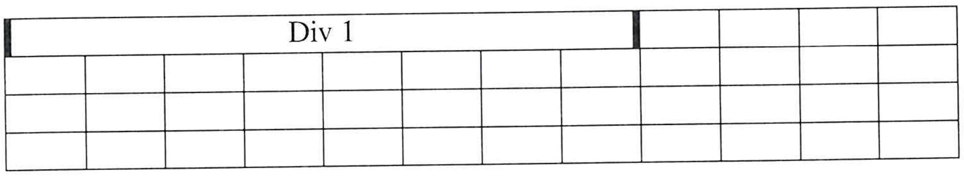


Figure 8: Small browser window – 530px

```
<!doctype html>
<html lang="en">

<head>
  <meta name="viewport" content="width=device-width, initial-
      scale=1 shrink-to-fit=no">
  <!-- Assume Bootstrap is imported -->
  <style>
      div div div { border: 1px solid black; }
  </style>
</head>

<body>
  <div class="container-fluid text-center">
    <div class="row">
      <div class="col-8 col-sm-4">Div 1</div>
      <div class="col-3 col-md-2">Div 2</div>
      <div class="col-2 col-md-1">Div 3</div>
    </div>

    <div class="row">
      <div class="col-5 col-md">Div 4</div>
      <div class="col-2 col-lg">Div 5</div>
      <div class="col-5 col-md">Div 6</div>
    </div>
  </div>
</body>
</html>
```

Figure 9: Responsive layout example

6 Communication Protocols (5P)

Figure 10 shows an HTTP request. Name the respective parts of the request marked with an arrow (A to E).

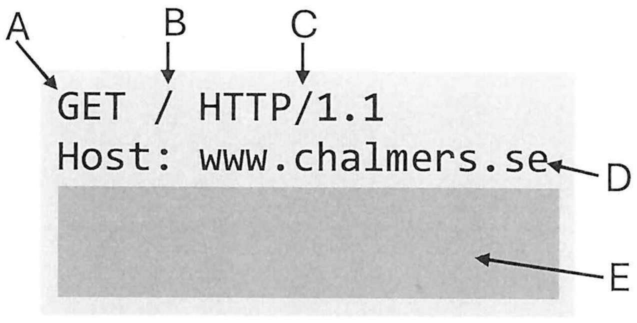


Figure 10: HTTP request

7 Authentication (3P)

Name three authentication methods.

8 Client-side vs. Server-side languages (4P)

Describe the difference between client-side and server-side languages. Address in particular the execution time and place.

9 JavaScript (6P)

Figure 11 shows an example `Stop`-object using JSON. Write a constructor function in JavaScript that creates a `Stop`-object and sets the values for the two attributes `name` and `platform`.

```
{name: "Stenpiren",
platform: "D"}
```

Figure 11: JSON representation of an example `Stop`-object