CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn		
DIT341	Web development		
Anonymous code Anonym kod	Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
1316	21/10/25	10	VG

* I confirm that I've no mobile or other similar electronic equipment available during the examination.
Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under eximinationen.

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	x	17	
2	x	19	
3	x	27	
4	x	8	
5	x	8	
6	x	6	
7	x	8	
8	x	6	
9			
10			
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			

[redacted]

3

CHALMERS

Anonym kod 1316

Poäng på uppgiften (ifylles av lärare)

Löpande sid nr 1

Question no. Uppgift nr 1

1 8. post ✓

```
9-13  var newUser = {
          name: req.body.name,
          role: req.body.role };
      users.push(newUser);
      res.status(201).json({id: users.length-1});
```

✓ 5

16 patch ✓

```
17-21  ~~var updatedUser = {~~
       ~~name:~~
       var id = req.params.user;
       var updatedUser = {
           name: (users[id].name || req.body.name)
           role: (users[id].role || req.body.role)};
       users[id] = updatedUser;
       res.status(201).json(updatedUser)
```

8

CHALMERS		(to be filled in by teacher)	Löpande sid nr 2
	Anonym kod 1316	Poäng på uppgiften (ifylles av lärare)	Question no. Uppgift nr 2

2 | (2.1) POST /canvas/courses

I would expect this API method to create a new course in Canvas. By using this method I should be able to pass a JSON in the request body (the structure of the JSON is predefined) and create a new course.

POST /canvas/courses/:course/assingments

I would expect this API method to create a new assignment within a specific course (e.g., defined by a course id). By using this method I should be able to pass a JSON in the request body and create a new assingment for ~~f~~ a specific course.

✓

Difference: The second method uses the relationship between a course and assingments. This gives us the opportunity to link an assignment to a course.
Prerequisite => we should know the course identifier (for instance, id) before hand.

(2.2) GET /canvas/users/1907 HTTP/1.1

host: www.canvas.com,
~~set-cookie: myCookie~~
......

~~[illegible]~~

2

3

CHALMERS

Anonym kod 1316	Poäng på uppgiften (ifylles av lärare)	Löpande sid nr 3	Question no. Uppgift nr 2

2.3 Status code 401 UNAUTHORISED

This would mean that the user trying to invoke the API call is unauthorized. It may occur when the user has not logged in to the system.

✓

Status code 403 FORBIDDEN

This would mean that the user lacks a special authorization (e.g. role) and, therefore, the call fails. The user is denied access to whatever he was trying to change.

2.4 I do not consider this as a good RESTful design as the method updates (patches) a specific user with a GET Request which is considered a safe one. But in reality we patch a user and this is ~~an~~ unsafe (or idempotent, depending on implementation)

Instead, the method can be implemented like this:

PATCH /canvas/users/:user

✓

The ~~B~~ backend should be ready to change any field that is sent with the request body.

3

CHALMERS

Anonym kod 1316	Poäng på uppgiften (ifylles av lärare)	Löpande sid nr 4
		Question no. Uppgift nr 2

2.5 Safe methods are those which do not make any changes to the DB when called. Unsafe ones are those which make changes to the DB each time they are called. Idempotent methods can be described as methods which at first call make a change but if tried to call the same method several times, a modification does not occur. PUT is a such HTTP method. First, the whole entity is changed but if invoked with the same body, a change does not happen.

GET /canvas/users → safe

GET /canvas/users/:user → safe

GET /canvas/users/:user?update=name&set=Leither

→ the browser assumes it is safe but in reality we make a change which makes it unsafe

POST /canvas/courses → unsafe

POST /canvas/courses/:course/assingments → unsafe

PUT /canvas/courses/:course → idempotent

← Example

✓

1 9

3

CHALMERS

Anonym kod 1316

Poäng på uppgiften (ifylles av lärare)

Löpande sid nr 3

Uppgift nr 3

3)

```
3  h1
4  div[green] > p
5  .label
6-9 @media screen and (max-width: 750px){
        ~~.first-button~~
        .button-wrapper {
            display: none;
        }
    }

27-30 <div class="button-wrapper">
        <p class="label"> First Button: </p>
        <button type="button" onClick="buttonPressed(1)">
            Button 1
        </button>
        <p class="label"> Second Button: </p>
        <button type="button" onClick="buttonPressed(2)">
            Button 2
        </button>
      </div>
```

✓

2 7

3

CHALMERS	Anonym kod	(to be filled in by teacher) Poäng på uppgiften (ifylles av lärare)	Löpande sid nr 6
	1316		Question no. Uppgift nr 4

4)

9 —

11 updateTeacherList() ✓

15-17

```
<tr v-for:"user in users" v-bind: ~~user.role~~ = "if (user.role === 'Teacher')
                                                    use.Teacher
                                                  " else
                                                    use.TA "
    <td>
        {{ user.name }} ✓
    </td>
    <td>
        {{user. role }} ✓
    </td>
</tr>
```

8

3

CHALMERS	Anonym kod	(to be filled in by teacher) Poäng på uppgiften (ifylles av lärare)	Löpande sid nr
	1316		7
			Question no. Uppgift nr 5

5)

1. Client - Server

Seperate client and server as their concerns are different

2. Stateless

The client's request shall be self-containing. The server shall not log client's state.

3. Cacheable

A RESTful design should enable caching to ~~improve~~ ensure better performance.

4. Uniform Interface

Changes to implementation should not affect the interface. Interface should be independent.

✓

CHALMERS	Anonym kod	Poäng på uppgiften (ifylles av lärare)	Löpande sid nr 0
	1316		Question no. Uppgift nr 6

6) If implemented as a „traditional" web app, any changes requested by the client would require the server to send a whole „new" page and client's browser would need to render all of it. On the other hand, if implemented as a Single Page Application, the communication ~~tha~~ between client and server would be via AJAX calls which follows the principle ~~of~~ „request and get what is ~~only~~ needed only".
For example, if the user wants to see a new assignment, using „traditional" principles would mean that ~~very~~ the server will send a whole new HTML page possibly with very few changes to the previous one. But if we use SPA, the client would only request what's needed, in our case a new assignment page, and the server would send only the data that needs to be changed on the page.

7) Cookies are mostly used to overcome the statelessness of the server. Usually the servers do not care about client's state but cookies are particularly useful for persistent authorization, meaning that via cookies the server would „remember" that the client has logged in. This could be particularly useful for our case, remember that the user has logged in so they ~~[illegible]~~ do not need to re-login on refresh. To ensure safety, the server could set an expiration date on a cookie (e.g. the user should confirm credentials after the cookie expires and if it's successful, a new expiration time is set).

Example: The user sends a request

```
GET /canvas/users/21 HTTP/1.1
host: www.canvas.com
~~cookie: null~~
```

The server verifies the user, sets a cookie and then sends it to the user along with the requested data

```
HTTP/1.1 200 OK
set-cookie: ~~session~~ cookieHash 1a 2a 3a
connection: keep-alive
......
content-type: JSON
[response body]
```

✓

CHALMERS

Anonym kod 1316

Poäng på uppgiften (ifylles av lärare)

Löpande sid nr 10

Question no. Uppgift nr 8

8) TCP

Used for communication between two computers. Once connection is established data can be sent both ways. QoS is ensured, loss packets are re-sent or at least an attempt is made.

Use case: 1-on-1 WhatsApp call connection. Data is constantly exchanged between the participants.

UDP

Used for communication between multiple computers where only one of them send data. The sender does not know who listens

Use case: Streaming -> Twitch streams, streamer's feed is sent to whoever's listening / watching