CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
DIT 342	Web development			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
572		2023-10-23	9	5

* 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	18	
2	X	19	
3	X	14	
4	X	9	
5	X	10	
6	X	9	
7	X	8	
8	X	10	
9			
10			
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			
Total examination points		97	

4

Question 1 Backend Development

```
12  app.patch('/pages/:name', function (req, res){
13-17:   const title = req.params.name
         for (i=0; i < pages.length ; i++){
           if (pages[i].title == title ){
             if (req.body.content ){
               pages[i].content = req.body.content
             } if (req.body.attachments ){
               pages[i].attachments = req.body.attachments
             }
             return res.status(200).json(pages[i])
           }
         }

20  app.post('/pages/:name/attachments', function (req, res){
21-25    const title = req.params.name
         for (i=0; i< pages.length ; i++ ){
           if ( pages[i].title == title ) {
         pages[i].attachments.push (req.body.attachments)
         return res.status(201).json({"attachments": pages[i].attachments})
       }}
```

4

Question 2 REST APIs

2.1

POST /wiki/pages
POST /wiki/pages/:page/attachments

I would expect the POST /wiki/pages to create a new page in the wiki. Once created, it would return code 201. If there was an error, it would most likely return 500 (internal server error)

The POST /wiki/pages/:page/attachments should create a new attachment to an already existing page. ~~If the method~~
The difference of these two is that this one already has a page in the wiki and is adding a new attachment. If successful, it should return code 201, if the page was not found, then 404. 3

multiple calls? req (&resp) content?

2.2

The problem with GET /wiki/visits is, that it wants to count the number of visits to a specific page. However, this endpoint does not specify a page. Then it should be
GET /wiki/pages/:page/visits

But this is not good either because in Rest GET is considered safe. Thus, it should not modify anything in the database.

Also, the browser might think that GET is cacheable and thus the count of visits might not be updated on every reload.

I think it would be better to use a POST request. that would every time increment the visit count to a page. 4
POST /wiki/pages/:page/visits

2.3 PATCH /wiki/dev? modification = "Raw MongoDB Query"

This design is not following the uniform interface constraint of REST. The constraint says that resources should be handled through representations. In this case, the resources are modified straight in the database. This imposes a security risk since developers could mess up the queries. Also, then anyone who writes this URI could modify the resources.
It also does not follow the constraint that each resource should be uniquely accessible? in this case you could access any resource with a plain database query. 5

breaks 3-tier arch

4

2.4

Safe - Safe operations are the ones that do not make any modifications to the resources.

Out of these 5, safe ~~are~~ is

GET /wiki/pages/:page

Idempotent - Idempotent operations are those that upon sending the exact same request multiple times, the first one does modifications to the resources/database, however, the next ones do not do further modifications.
Idempotent request take the resources to an expected state. All ~~of~~ safe methods are also idempotent

Out of thes 5, idempotent are:
~~(GET /wiki/pages/:page) - also safe~~
PUT /wiki/pages/:page
DELETE /wiki/pages/:page
+ the safe one from the top

Additional idempotent (but not safe) method:

DELETE /wiki/pages/:page/attachments

which would delete all attachments in a page. 7

CHALMERS

Anonymous code
Anonym kod
DIT342 572

Points for question (to be filled in by teacher)
Poäng på uppgiften (ifylles av lärare) 14

Consecutive page no. Löpande sid nr 4

Question no. Uppgift nr 3

Question 3

15 - orange
16 - black
17 - yellow
18 - black
20 - green
22 - purple
23 - pink

4

Question 4 Frontend Development

7 <input v-model = " new Title " >

8 <button @click = " rename Page() " > Rename the page </button>

~~27-30~~

1a

27-30 ~~axios.patch(/wiki/pages/ {{this.title}}, this.newTitle)~~

~~axios.~~
~~axios.patch(/wiki/pages + {{this.title}}), function(req, res), this.newTitle~~

~~}~~

27-30

axios.patch(/wiki/pages + {{this.title}}, this.newTitle)"

as json?

~~this.title = this.newTitle~~

```
if (res.status == 200){
    this.title = this.new Title
    this.new Title = ''
}
```

NB! In figure 2, the available methods only had PUT for modifying a page. However in this case we do not change every attribute of a page, only the name. Thus I use patch which was not one of the methods in figure 2.

4

Question 5

First a user types the URL in their browser:

```
GET  HTTP/1.1
Host: dit342-exam.se
Accept: text/html
```

The browser loads the page:

```
HTTP/1.1  200 OK
Host: dit342-exam.se
Content-Type: text/html

<!doctype html>
<html>
<head>
<title> TITLE </title>
</head>  <body> ~~Hello </body>~~ <button> click </button> </body>
</html>
```

Then the user click on a button. This calls the axios and AJAX libraries. These call the following HTTP method:

```
GET /camels  HTTP/1.1
Host: dit342-exam.se
Accept: application/json
```

And will get a response: (assuming it is successful)

```
HTTP/1.1  200 OK
Host: dit342-exam..se
Content-Type: application/json

{"name": "Camel", "age": "2"; "name": Camel2", "age": "3"}
```

And then using Vue.js, the list of camels will be shown on the screen

If we had cookies implemented, the first GET could have a header

Set-Cookie: session=1

And this cookie would be used in the other get request as

Cookie: session=1

But I did not include those since it was not mentioned

4

Question 6 RWD

Responsive Web Design means to design a single frontend that ~~we~~ could be used on devices with different screen sizes (phone and laptop for example) and on differend devices (also Braille devices). ~~Also, responsiv~~

Instead of creating multiple frontends such as one for computer screens, and a native app for Android, RWD's idea is to create one single. This is much cheaper and does not exclude some user groups.

How ~~does~~ can RWD be used to implement accessibility?

1) Use media queries for hiding purely visual content for screen-readers. Or use media queries like "speach" to tell screen readers what to "say"

2) RWD can be used to increase/decrease font size and/or contrast to make the text more visual for visually impaired users.

Two examples of providing web accessibility independently of RWD:

1) Using alternative text for images.
this can be done via the alt="" on an image.

2) Avoid time sensitive content such as pop-ups for small amount of time. ~~This also includes~~ ~~Had you should avoid small clickable items to close~~

4

Question 7 Testing

Before deployment:

1) Integration test - You could test the new feature with integration test to make sure that the backend and database are working together correctly. This can be done using Postman.

2) System test - Testing the system as a whole. This is usually done manually. You can test if the whole system works together as should.

After deployment:

1) Canary release - You could release the new feature to only a small number of users, let's say 5%. The other 95% would use the old version but you have the 5% test group to see if it works as intended.

2) A/B testing - ~~Let's say the new feature also includes a new frontend feature~~. You could have 2 versions of the new feature and let 50% of users use one of the versions and 50% to use the other in order to see which one performs better.

4

Question 8

TCP vs UDP

TCP - ~~Transport~~ Transmission control Protocol

TCP is a protocol used in OSI layers 4 and 5. it's a protocol that enables a stable connection between two "users". Usually used to transport data over the network.

TCP is reliable and ordered which means that it makes sure the data is delivered in the same order as it was sent and tries to recover lost data.

In TCP, two "users" can communicate with eachother in both directions.

it is used for emailing, messagin, web browsing. TCP could be slower than UDP since it is reliable and ordered.

UDP - User Datagram Protocol

UDP is also used in OSI 4 & 5 layers and used for transporting data over the network.

UDP has one sender and many receivers. The sender does not necessarily know who is the receiver or if they actually receive the data. also supports unicast

It is not ordered nor reliable in a sense that UDP does not try to recover lost data.

UDP is thus faster than TCP which is why it is often used for streaming services or also online gaming.