

TDP003 Projekt: Egna datormiljön

Testdokumentation

Författare

Jimmie Roos, jimro697@student.liu.se
Sebastian Grunditz, sebgr273@student.liu.se

Innehåll

1	Introduktion	1
1.1	Testmetod	1
2	Presentationslagret	1
2.1	Kravlista	1
2.2	Index	3
2.3	List	3
2.4	Techniques	4
2.5	Project	4
2.6	Allmän 404	5
3	Log-fil	6
4	Datalagret	7
4.1	Load	7
4.2	Get project count	7
4.3	Get project	7
4.4	Get latest project	8
4.5	Search	8
4.6	Search project	9
4.7	Check search	9
4.8	Get techniques	9
4.9	Get technique stats	10

Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Första utkast	22/10-18

1 Introduktion

Detta är vår testdokumentation för projektet i TDP003 där vi beskriver hur vi och andra har testat vår hemsida och vilka fel som uppkommit. Vi beskriver även hur vi hanterat dessa fel och löst dem.

1.1 Testmetod

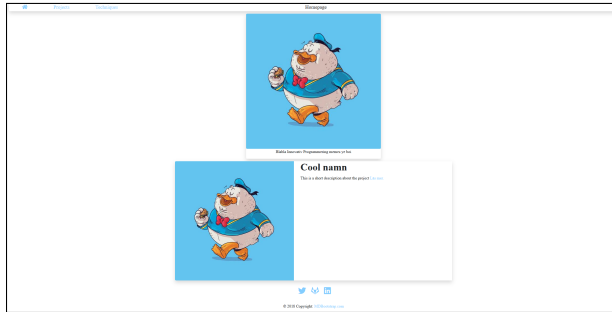
Vi har själva testat hemsidan under utveckling med olika scenarion för att se att koden funkar som den ska. Under torsdagen 18/10 så hade vi systemdemonstration för övriga i klassen där dom fick testa att ha sönder vår sida och hitta fel. Vi skrev ner all feedback vi fick för att kunna fixa dem och dokumentera dem i detta dokument tillsammans med lösningarna.

2 Presentationslagret

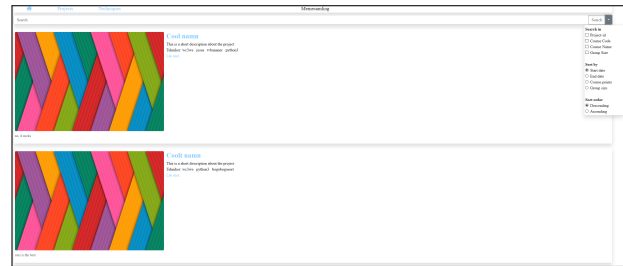
2.1 Kravlista

- Förstasida med bilder. URL: /. Se figur 1a.
- Söksida som visar en lista över projekt med kort information om varje projekt och som gör det möjligt att sortera dessa, samt söka bland dem genom ett formulär på sidan. Url: /list. Se figur 1b.
- Projektsida som visar fullständig information om ett projekt. GET variabel för att ange projekt-id: id URL: /project/id - där id är projektets nummer. Se figur 1c.
- Tekniksida som visar information om alla projekt utifrån använda tekniker. URL: /techniques. Se figur 1d.
- För varje projekt ska en liten bild visas på söksidan och en stor på projektsidan. Det behöver inte vara samma bild. Bildtext för varje bild skall finnas. Se figurer 1b och 1c.
- Vid fel ska systemet skriva ut informativa meddelanden till användaren på en lämplig nivå för en slutanvändare. (Det vill säga, systemet ska fånga och omvandla felkoder och statuskoder till begripliga meddelanden.). Se figur 4.
- När en användare försöker visa ett projekt som inte finns, ska korrekt statuskod returneras (dvs. 404). Se figurer 1e och 1f.

2.1.1 Bilder för kravlista



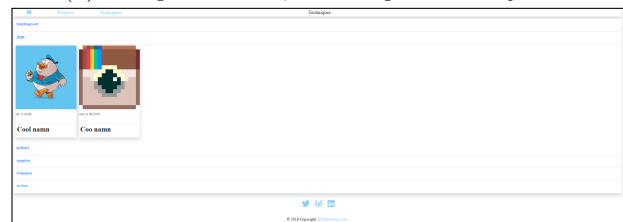
(a) Bild på startsidan



(b) Bild på söksidan, med *dropdown-meny* nere



(c) Bild på hur en projectsida ser ut



(d) Tekniksidan, med en teknik ”öppen”

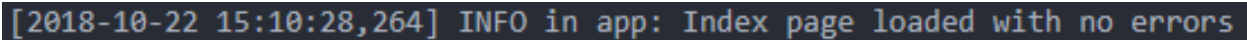


(e) 404-sidan med relevant information, och en lustig bild



(f) 500-sidan med informativ text om vad som hänt.

2.2 Index

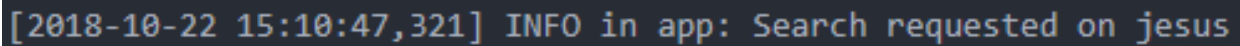


```
[2018-10-22 15:10:28,264] INFO in app: Index page loaded with no errors
```

Figur 2: Lyckad laddning av start-sidan

Sidan uppfyller alla krav som är listade under kravspecifikation i systemspecifikationen. Har ej upptäckt några fel under testkörningar eller systemdemonstration.

2.3 List



```
[2018-10-22 15:10:47,321] INFO in app: Search requested on jesu*****
```

Figur 3: Lyckad sökning på project-list-sidan

Alla krav i systemspecifikation har blivit mötta. Under systemdemonstrationen så stötte vi på problemet att vi inte kunde söka i tekniker. Det åtgärdade vi genom att ändra i datalagret så att istället för att returnera resultatet av ett rekursivt anrop på Check search, se bild 1, så returnerar vi True om rekursivt anrop på Check search returnerar True, se bild 2.

```
1 return check_search(sökord, lista)
```

Listing 1: retur av rekursivt anrop

```
1 if check_search(sökord, lista):  
    return True
```

Listing 2: Retur True om check_search()

2.3.1 Fel i sökning

```
[2018-10-23 14:59:44,309] ERROR in app: Exception on /list [GET]
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 2292, in wsgi_app
    response = self.full_dispatch_request()
  File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1815, in full_dispatch_request
    rv = self.handle_user_exception(e)
  File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1718, in handle_user_exception
    reraise(exc_type, exc_value, tb)
  File "/usr/local/lib/python3.6/dist-packages/flask/_compat.py", line 35, in reraise
    raise value
  File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1813, in full_dispatch_request
    rv = self.dispatch_request()
  File "/usr/local/lib/python3.6/dist-packages/flask/app.py", line 1799, in dispatch_request
    return self.view_functions[rule.endpoint](**req.view_args)
  File "/mnt/e/Ip/TDP003/projekt/app.py", line 78, in list
    ditem = json.loads(item)
  File "/usr/lib/python3.6/json/_init_.py", line 354, in loads
    return _default_decoder.decode(s)
  File "/usr/lib/python3.6/json/decoder.py", line 339, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  File "/usr/lib/python3.6/json/decoder.py", line 355, in raw_decode
    obj, end = self.scan_once(s, idx)
json.decoder.JSONDecodeError: Expecting ':' delimiter: line 1 column 8 (char 7)
[2018-10-23 14:59:44,319] ERROR in app: Error in database, check above for more info
```

Figur 4: Någon har mixtrat i url:en

Om man ändrar i url:en i "http:.../list" så kastas det en 500 server error och man kommer till en sida som tar hand om *500-error*.

2.4 Techniques

```
[2018-10-22 15:11:11,974] INFO in app: /Techniques loaded.
```

Figur 5: Lyckad laddning av tekiksidan.

Alla krav för tekniksidan har blivit mötta. Har ej hittat några fel under tester eller systemdemonstration.

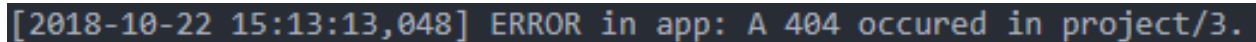
2.5 Project

```
[2018-10-22 15:12:58,210] INFO in app: Project 1 loaded
```

Figur 6: Lyckad laddning av projektsida.

Omdirigering från annan sida kommer leda till lyckad laddning av "http:../project/<id>"

2.5.1 Project 404



```
[2018-10-22 15:13:13,048] ERROR in app: A 404 occurred in project/3.
```

Figur 7: Försökte gå in på ett projekt som inte existerar.

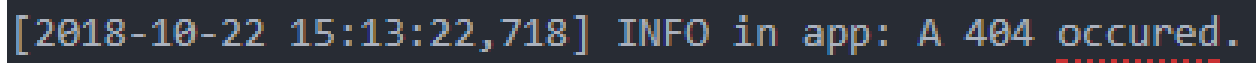
Under systemdemonstrationen så märkte vi att det kastades en 500 error om vi försökte komma in på ett projekt som inte existerade genom att ändra i url:en. Det fixade vi genom att lägga till:

```
1 if project == None:
2     app.logger.error('A 404 occurred in project/{0}'.format(project_id))
3     return render_template('page_not_found.html', page_name="Uh oh")
```

Listing 3: Kollar om projekt existerar.

Detta kollar om projektet vi försöker gå in på existerar i databasen och om det inte gör det så kastar vi en 404 error och laddar 404 sidan, se fig 7. Detta går att repetera enkelt, genom att i url:en ändra så att det står "http:../project/<id>" där <id> är id på ett projekt som inte finns.

2.6 Allmän 404



```
[2018-10-22 15:13:22,718] INFO in app: A 404 occurred.
```

Figur 8: Försök att gå in på sida som inte existerar.

Om vi försöker gå in på en sida som inte existerar, till exempel "http:../hej", så kommer vi bli skickade till en 404 sida istället och kasta en error till debug.log, se fig 8.

3 Log-fil

```
1 [2018-10-22 16:41:01,148] INFO in app: Current systemversion: V.1.0.1
2 [2018-10-22 16:41:05,735] INFO in app: Index page loaded with no errors
3 [2018-10-22 16:41:15,683] INFO in app: Search page loaded with no errors
4 [2018-10-22 16:41:30,510] INFO in app: Search requested on python
5 [2018-10-22 16:41:37,852] INFO in app: Project 2 loaded with no errors
6 [2018-10-22 16:41:42,829] INFO in app: Techniques page loaded with no errors.
7 [2018-10-22 16:41:51,461] ERROR in app: A 404 occurred in project/3.
8 [2018-10-22 16:42:05,097] ERROR in app: A 404 occurred.
9 [2018-10-22 16:42:56,241] INFO in app: Search page loaded with no errors
```

Listing 4: Bit av en log-fil

I figur 4 visar vi en kontrollerad version av hur en "debug.log" fil. Detta för att visa vilka log-meddelande vi har skickat in i log-filen, från vårt presentationslager. I ordning uppifrån och ner händer följande:

- Nuvarande version av hemsidan skrivs ut.
- Index-sidan laddas utan problem.
- Sök-sidan laddas utan problem.
- Vi söker med *python*, och inga problem uppstår under sökning.
- "http:.../project/2" laddas utan problem.
- Tekniksidan laddas utan problem.
- Försök att ladda "http:.../project/3", vilket inte finns, så ett 404-error kastas
- Försök att ladda en sida som inte finns, till exempel "http:.../hej", som inte finns, så ett 404-error kastas
- Sök-sidan laddas utan problem

Det som är bra med loggen, är att den beskriver exakt när problemet uppstår, både i datum och klockslag. Detta underlättar ytterligare för att veta vilken version av hemsidan det var som krashade.

4 Datalagret

För fullständig information om dessa funktioner, se [kravspecifikationen](#)

4.1 Load

4.1.1 Indata

Filnamn i *string*-format till en ".json"-fil som ska laddas in och visas på sidan.

```
1 def load(filename):
```

Listing 5: Definering av **load()** funktionen

4.1.2 Förväntad utdata

En *list*-representation av den databas man vill ladda in, enligt hur **json.load(*fil*)** laddar in *.json*-filer.

4.1.3 Möjliga fel

Om filen inte finns, returnerar **load** *None*, vilket resulterar i att sidan kraschar och man skickas till en *500-error* sida, så att man får lite information om felet.

```
1 except FileNotFoundError:  
    return None
```

Listing 6: Hantering när den inte hittar filen

4.2 Get project count

4.2.1 Indata

Databasen som vi laddar med Load representerad av en *list*.

```
def get_project_count(db):
```

Listing 7: Definering av **get_project_count()** funktionen.

4.2.2 Förväntad utdata

Antalet projekt i databasen.

4.2.3 Möjliga fel

Vi har inte hittat något som kan gå fel i denna funktion.

4.3 Get project

4.3.1 Indata

Skicka in en databas, efter att den laddats med hjälp av Load, och ett projekt-id på formen *integer*.

```
1 def get_project(db, id):
```

Listing 8: Definering av **get_project()** funktionen.

4.3.2 Förväntad utdata

Ett projekt så som det representeras i databasen, i vårt fall är det en *dictionary* med data i.

4.3.3 Möjliga fel

Om project-id inte finns i database, skickas en *None-type*.

```
1 return None
```

Listing 9: Retur ifall inget projekt hittas

4.4 Get latest project

4.4.1 Indata

Skickar in databasen som laddas med Load. Projekten sorteras efter datum med hjälp av Search för att få det senast skapade projektet.

```
1 def get_latest_project(db):
```

Listing 10: Definering av `get_latest_project()` funktionen.

4.4.2 Förväntad utdata

Det senaste projektet ur databasen på samma format som det existerar i databasen.

4.4.3 Möjliga fel

Om datumformatet på projektet inte stämmer överens med det formatet som vi använder så kommer inte projektet att sorteras på rätt sätt. Då kommer de att komma i fel ordning.

4.5 Search

Använder sig av två underfunktioner Search project och Check search.

4.5.1 Indata

`search()` tar in mycket data, så de listas nedan, med en kort beskrivning.

- `db` -> Databas, som fås från Load.
- `sort_by` -> En *string* som bestämmer vilket datafält man sorterar på.
- `sort_order` -> En *string* som bestämmer om man sorterar efter "stigande" eller "fallande".
- `techniques` -> En *lista* med tekniker som man vill söka efter i projekten.
- `search` -> Fritext på *string*-format.
- `searchfield` -> En *lista* med *string*, som bestämmer i vilka datafält man söker med "search"

```
1 def search(db, sort_by='start_date', sort_order='desc',  
    techniques=None, search=None, searchfield=None):
```

Listing 11: Definering av `search()` funktionen.

4.5.2 Förväntad utdata

Databasen filtrerad på de sökkriterier som skickats med, sorterad i antingen "stigande" eller "fallande".

4.5.3 Möjliga fel

Om Load returnerar *None* kommer Search att krascha. Kolla på Fel i sökning i Presentationslagret för mer info om vad som händer då.

4.6 Search project

4.6.1 Indata

Tar in ett projekt representerat av en *dictionary*, ett sökord representerat av en *string* och ett sökfälts-filter representerat av en *string*, om man användaren vill filtrera på det.

```
def search_project(project, search_word, searchfield):
```

Listing 12: Definering av search_project()

4.6.2 Förväntad utdata

Om sökordet finns i projektet, returnerar den *True*, annars *False*.

4.6.3 Möjliga fel

Då denna funktion fungerar som en mellanhand mellan Search och Check search, kastas det inga fel i denna funktion.

4.7 Check search

4.7.1 Indata

Tar in ett sökord, representerat av en *string*, och ett datafält, representerat av en *list*, en *integer*, eller en *string*, för att söka i.

```
1 def check_search(search_word, item):
```

Listing 13: Definering av check_search() funktionen.

4.7.2 Förväntad utdata

Antingen så returnerar vi *True* eller *False* beroende på om sökordet matchar datafältet eller inte.

4.7.3 Möjliga fel

Om vi skulle få in en *dictionary* som datafält så kommer det krascha då vi inte hanterar detta fall.

4.8 Get techniques

4.8.1 Indata

Skickar in databasen som laddas med Load för att sedan ta ut alla unika tekniker som används i projekten.

```
1 def get_techniques(db):
```

Listing 14: Definering av get_techniques()

4.8.2 Förväntad utdata

En sorterad lista med alla unika tekniker som använts i projekten i databasen.

4.8.3 Möjliga fel

Vi har ej hittat några möjliga fel under testerna av denna funktion. Blir något fel så händer det innan denna funktion kan anropas.

4.9 Get technique stats

4.9.1 Indata

Skickar in databasen och alla tekniker vi får ut från Get techniques för att sedan sortera vilka tekniker som används för vilka projekt.

```
1 def get_technique_stats(db):
```

Listing 15: Definering av get_technique_stats()

4.9.2 Förväntad utdata

En dictionary med tekniker som nyckel med projekten som använder tekniken som item till nyckeln i en lista.

4.9.3 Möjliga fel

Vi lägger endast till tekniker som redan blivit filtrerade i Get techniques så här har vi inte hittat några möjliga fel.

4.9.4 Avvikelse från specifikation

`get_technique_stats()` avviker från specifikationen i det faktum att vi i listan för varje teknik lägger till hela projektet, istället för bara projektnamn och projektid. Detta har vi ändrat för att förenkla för oss på Techniques, så att vi inte behöver skicka med för många variabler. Vår kod ser ut som i figur 15, medan koden i figur 16 är enligt kravspecifikation.

```
1 if technique in item["techniques_used"]  
    technique_stats[technique].append(item)
```

Listing 16: Vår lösning på get_technique_stats().

```
1 if technique in item["techniques_used"]:  
2     technique_stats[technique].append({"id":item["project_id"], "name":item["project_name"]})
```

Listing 17: Lösningen enligt kravspecifikationen.