

10.2C: Search a Database

Task

In this task, you are required to extend your web server code from **Task 10-1P** to allow a user to search the **dKin Membership** database for entries matching a particular input from one of the database fields.

You will need to extend the main feedback form page (from **Task 10.1P**) by adding a **new form**. This will have two input fields, one for the *Search Term* and the other with a **pulldown select** for the *database field* to search in. A sample of a very simple *form & submit button* (added to the end of the main page) is shown in the screenshot below:

The screenshot shows a web browser window with the title 'dKin Membership' and the address 'localhost:3000'. The page content includes:

- A 'Comments' section with a text input field containing the placeholder 'Please provide a few words to describe your favourite cap...' and two buttons: 'Submit' and 'Reset'.
- A 'List Feedback' section with the text: 'The following button will issue a GET request to the /membershipdetails route to Retrieve all the members details stored in the DB.' Below this is a green button labeled 'Retrieve Membership Details From Database'.
- A 'Search Members Database' section with the text: 'The following button will issue a POST request to the /search route to retrieve rows from the dKin Caps database based on a given input field.' Below this is a form with two input fields: 'Search term:' with the placeholder 'Enter a term to search' and 'Field:' with a pulldown menu showing 'Table field...'. Below these fields is a green button labeled 'Search Membership Database'.

Task10.2.1 Form page with search field

In the example used here, the database holds the records of 9 members.

Membership List

localhost:3000/membershipdetails

dKin Caps

A young and inspiring organisation whose primary purpose is to bring you the highest quality headwear available.

Membership List

The table below lists all the registered members of the dKin Caps community. Accessing this data, stored in a persistant database, is performed on the server.

ID	Firstname	Surname	Email	Mobile#	Cap Type	Number of Caps	Comment
1	Mick	Hobbs	mick@deakin.edu.au	0499000111	Between 1 and 10 caps	Snapback Cap	I wash my cap 3 times a week!
2	Sam	Smith	samsmith@deakin.edu.au	0411987987	More than 30 caps	Sun Cap	Never go outside in the sun with out my trusty cap.
3	Alexander	Nguyen	aln@deakin.edu.au	0424366221	Between 1 and 10 caps	Runners Cap	I wash my cap at least 7 times a week... I don't like dirty caps.
4	Sophia	Taylor	sophie@deakin.edu.au	0433775775	0 caps	Bucket Hat	Yet to get a cap... But when I do the first will be a Blue Bucket Hap
5	Noah	Rodriguez	noah@deakin.edu.au	0462421133	30+ caps	Trucker Cap	Sometimes, wearing a trucker's cap gives me a feeling of anonymity or privacy, helping me blend in or go unnoticed in a crowd...
6	Derick	Long	dericklong@deakin.edu.au	0499111000	1 - 10 caps	Trucker Cap	I love my trucker cap. I wear it in the morning when I go to work as a barista.
7	Petra	Smithton	pppsss@deakin.edu.au	0400000001	30+ caps	Beanie	My beanie is warm and fluffy! I love my beanie.
8	Astrid	Lawson	starz@deakin.edu.au	0455666999	Between 1 and 10 caps	Sun Cap	I spend lots of time outside, so my trusty sun cap is always on my head. Gotta be sun smart!
9	Brodie	Hammersmith	hammer@deakin.edu.au	04123545678	1 - 10 caps	Snapback Cap	One cap... I love it... but I never, ever wash it.

[Click here to return to the feedback form.](#)

© 2024 A webpage footer with important info!

Task10.2.2 Databse containing 9 members records

The example below shows a search on term smith within the *Surname* field:

dKin Membership

localhost:3000

Search Members Database

The following button will issue a `POST` request to the `/search` route to retrieve rows from the *dKin Caps* database based on a given input field.

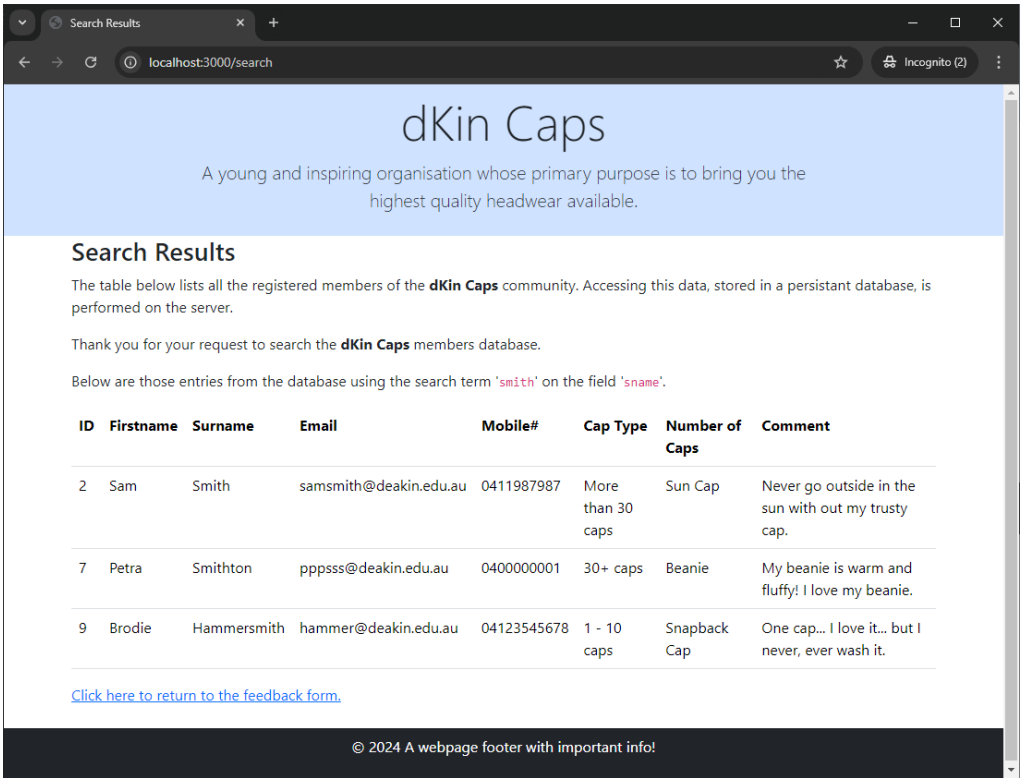
Search term: smith

Field: Surname

Search Membership Database

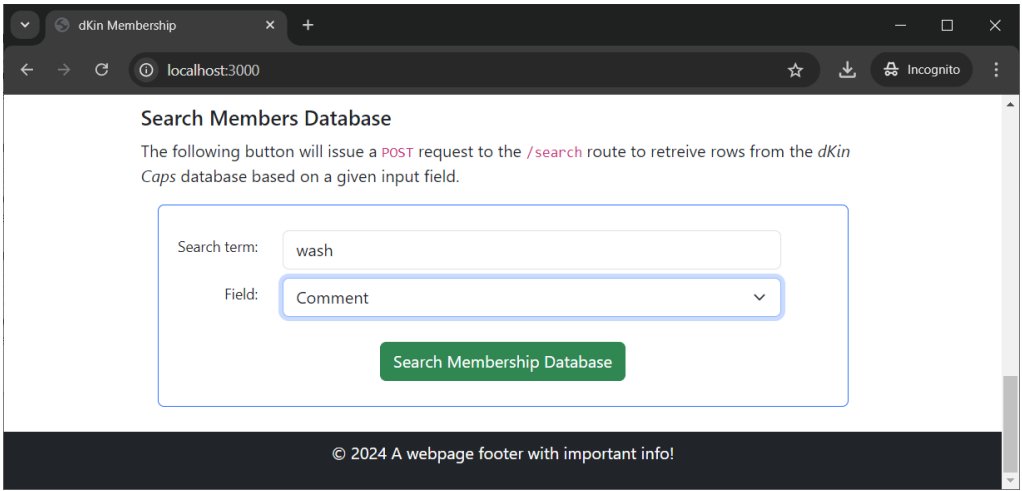
© 2024 A webpage footer with important info!

Task10.2.3 Input search field "smith"



Task10.2.3a Results of field "smith"

While the next example shows a search on term wash within the *Comment* field:



Task10.2.4 Input searching for "wash"

dKin Caps
A young and inspiring organisation whose primary purpose is to bring you the highest quality headwear available.

Search Results

The table below lists all the registered members of the **dKin Caps** community. Accessing this data, stored in a persistent database, is performed on the server.

Thank you for your request to search the **dKin Caps** members database.

Below are those entries from the database using the search term 'wash' on the field 'comment'.

ID	Firstname	Surname	Email	Mobile#	Cap Type	Number of Caps	Comment
1	Mick	Hobbs	mick@deakin.edu.au	0499000111	Between 1 and 10 caps	Snapback Cap	I wash my cap 3 times a week!
3	Alexander	Nguyen	aln@deakin.edu.au	0424366221	Between 1 and 10 caps	Runners Cap	I wash my cap at least 7 times a week... I don't like dirty caps.
9	Brodie	Hammersmith	hammer@deakin.edu.au	04123545678	1 - 10 caps	Snapback Cap	One cap... I love it... but I never, ever wash it.

[Click here to return to the feedback form.](#)

© 2024 A webpage footer with important info!

Task10.2.4a Results searching for "wash"

Note that in these examples the 'partial' word is used in the search. This is implemented using SQL commands, and as such it is possible to use wild cards. Here the search term is wrapped up in the % symbols, such as a search for smith in the *surname* would be %smith% and would return 3 members details.

Also, if a search term doesn't return any rows, a message like "Nothing found" or "List is empty" should be displayed.

Steps

To complete this task, you are required to:

1. Extend your `index.ejs` template to hold another form for the **Search Term** and **Field to Search**. The input fields should have a unique/relevant name such that the server can extract this from the request message. The form should `POST` the request to the route `/search`.
2. Extend your web server code (`index.js`) to handle a new `POST` request on the route `/search`. In this handler, the `search` data field should be extracted.
3. Issue multiple database request to find all rows that have a given *search term* (or substring) in various *fields* of the members database, like those examples provided above.
4. Create a new template file (`search.ejs`) that is called (`response.render()`) with the parameters: *title*, *search term* and *rows*. It should display the list of matching *rows* (as returned by the database request in step 3). **NOTE:** This template is very similar to the `members.ejs` template used in Task 10.1P.

Hints

The SQL code snippet for **Searching** the database on a field *like* a given term could be:

```
const query = `SELECT * FROM Comments WHERE ${searchfield} LIKE '${searchtype}'`;
```

Where the variables `searchfield` and `searchtype` is extracted from the *form* data passed into the handler.

What to submit

You should submit:

- Source code of the template web page of your extended main page *form* (i.e., the `index.ejs` file)
- Source code of the template file that renders the contents of the results of the database field search into a table (i.e., the `search.ejs` file)
- Source code of the *Node.js* server program file with the new `POST` handler for the `/search` route (i.e., the second `index.js` file).
- A document with **3 Screenshots** of the browser window showing at least 3 search terms:
 1. A specific *surname* that does exist
 2. A specific *comment term* that does NOT exist
 3. A search term that returns more than one row, from either the surname or comments fields