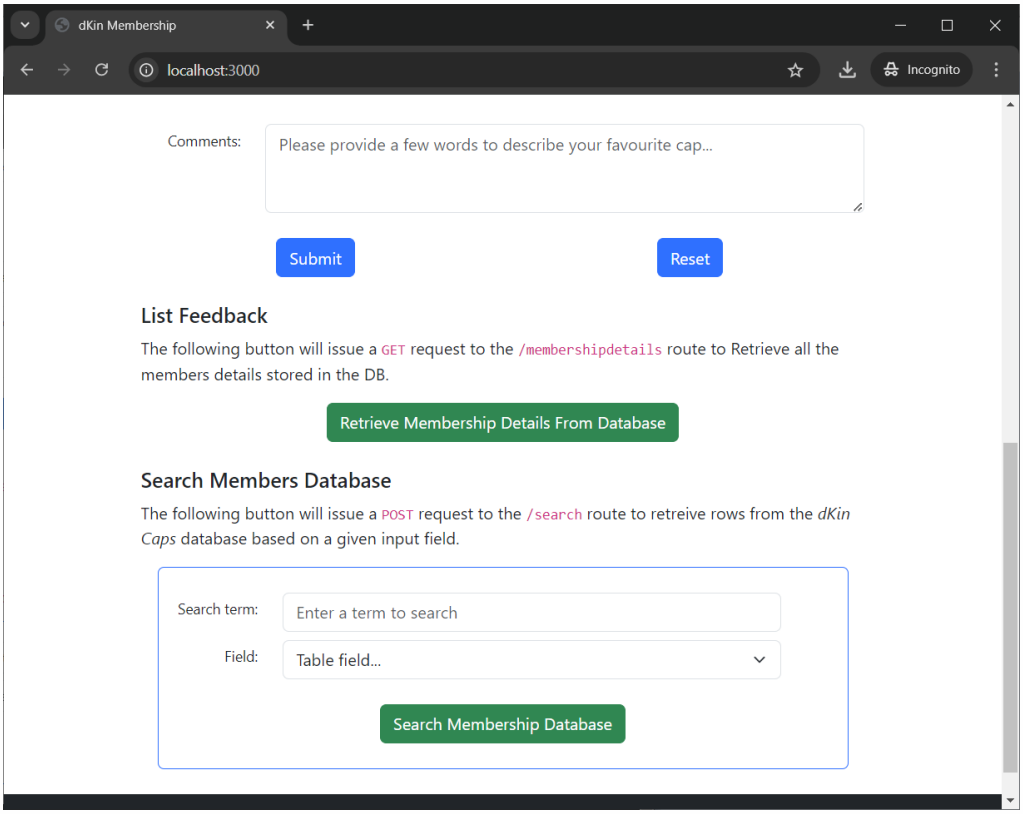
# 10.2C: Search a Database

## Task

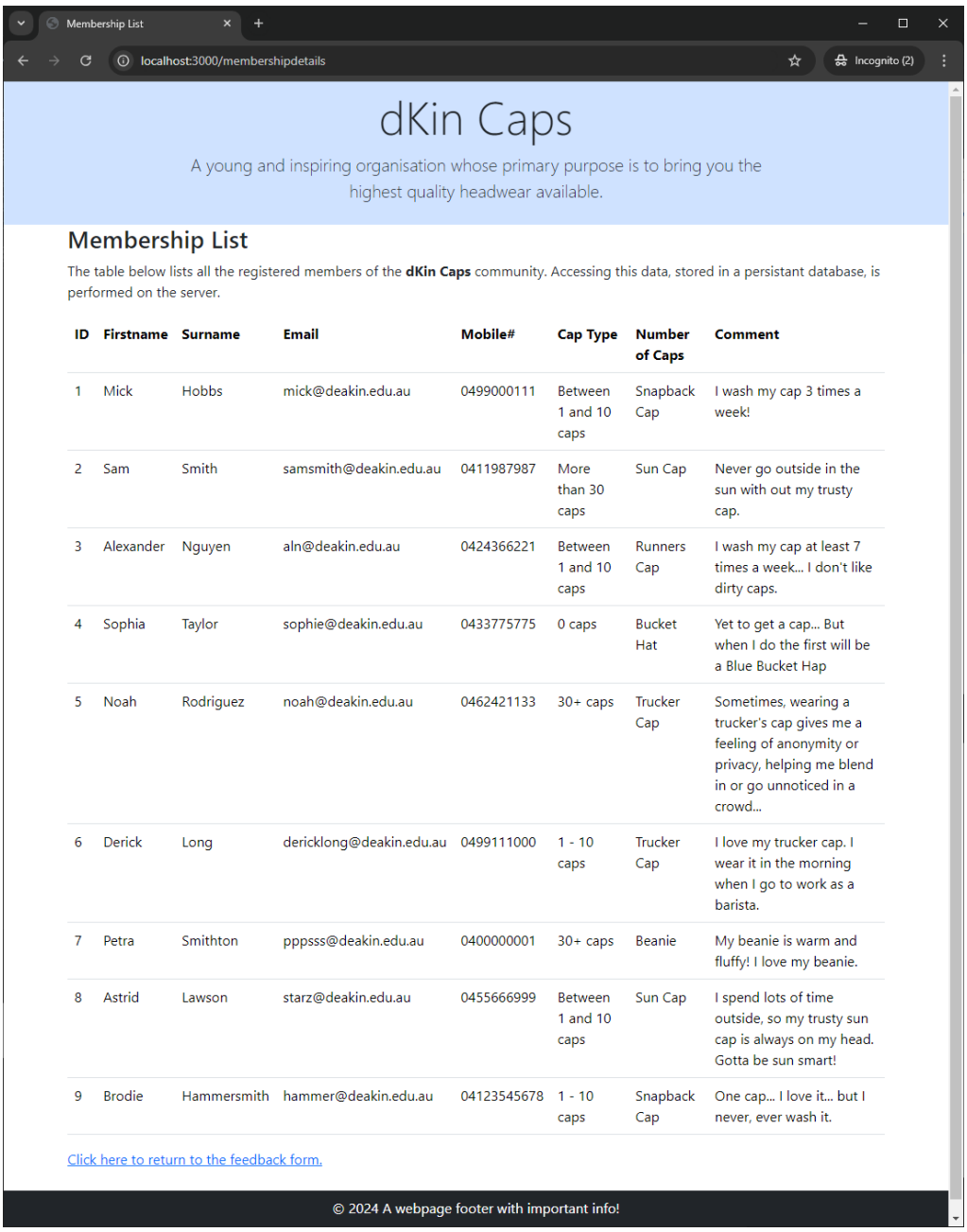
In this task, you are required to extend your web server code from **Task 10-1P** to allow a user to search the **dKin Membership** database for entries matching a particular input from one of the database fields.

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 *databse 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:



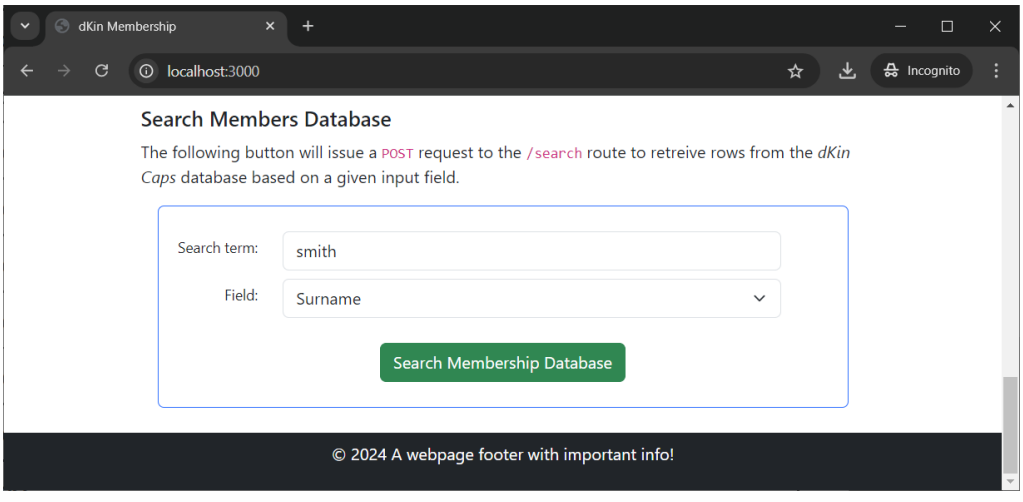

Task10.2.1 Form page with search field

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

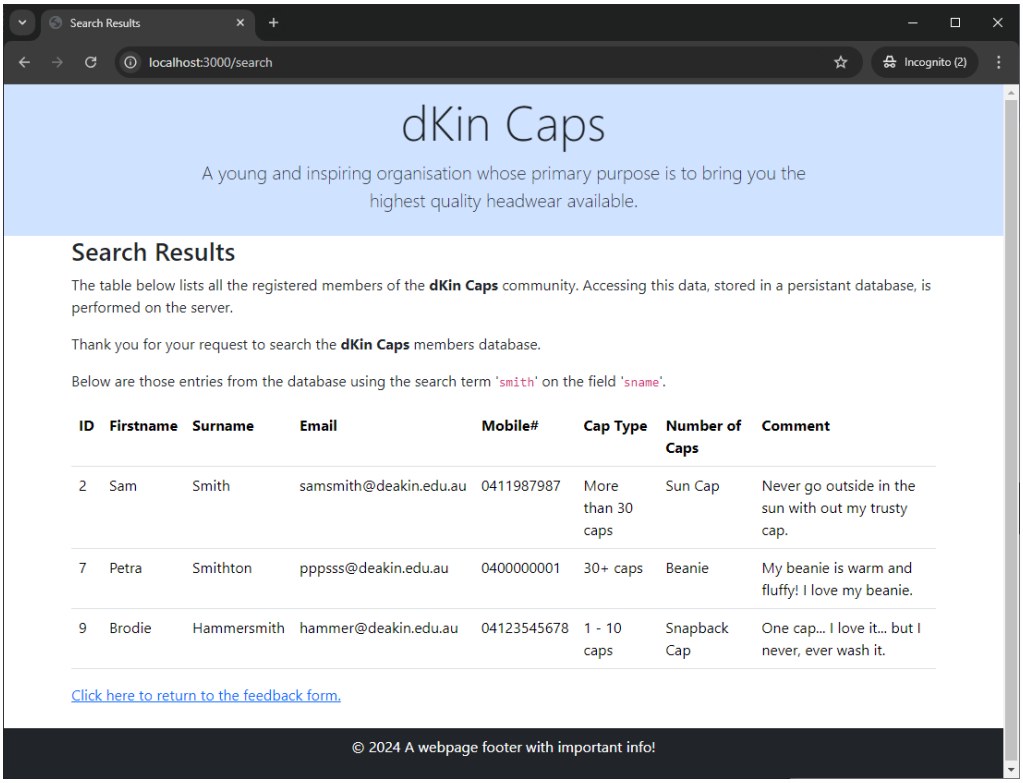



Task10.2.2 Databse containing 9 members records

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

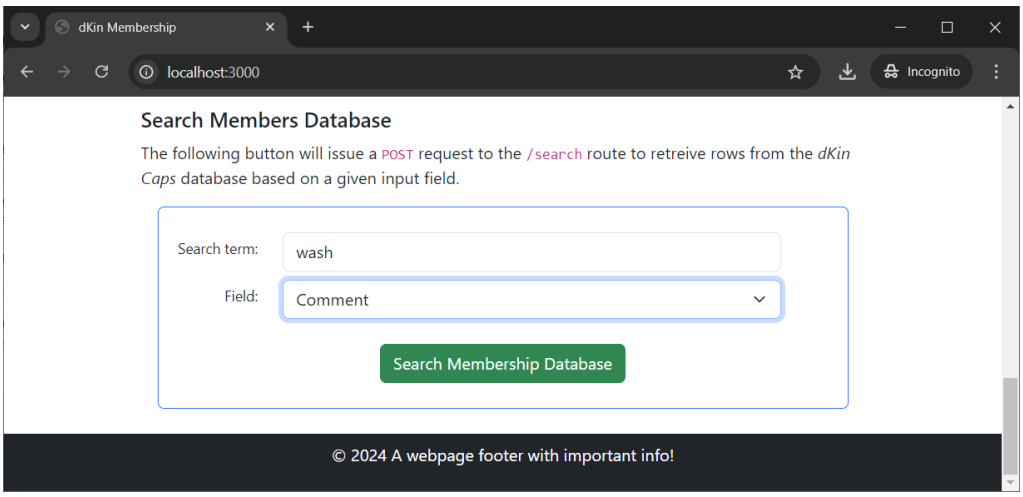


Task10.2.3 Input search field "smith"

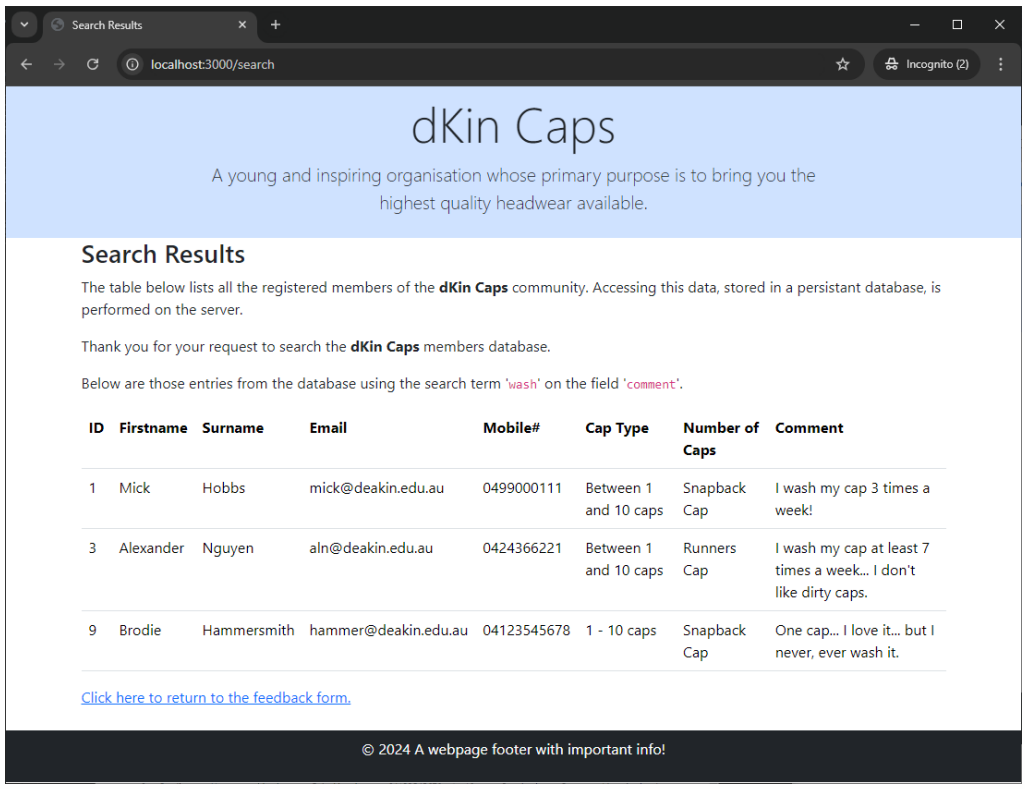



Task10.2.3a Results of field "smith"

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




Task10.2.4 Input searching for "wash"



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 as 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 varous *fields* of the members databse, 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