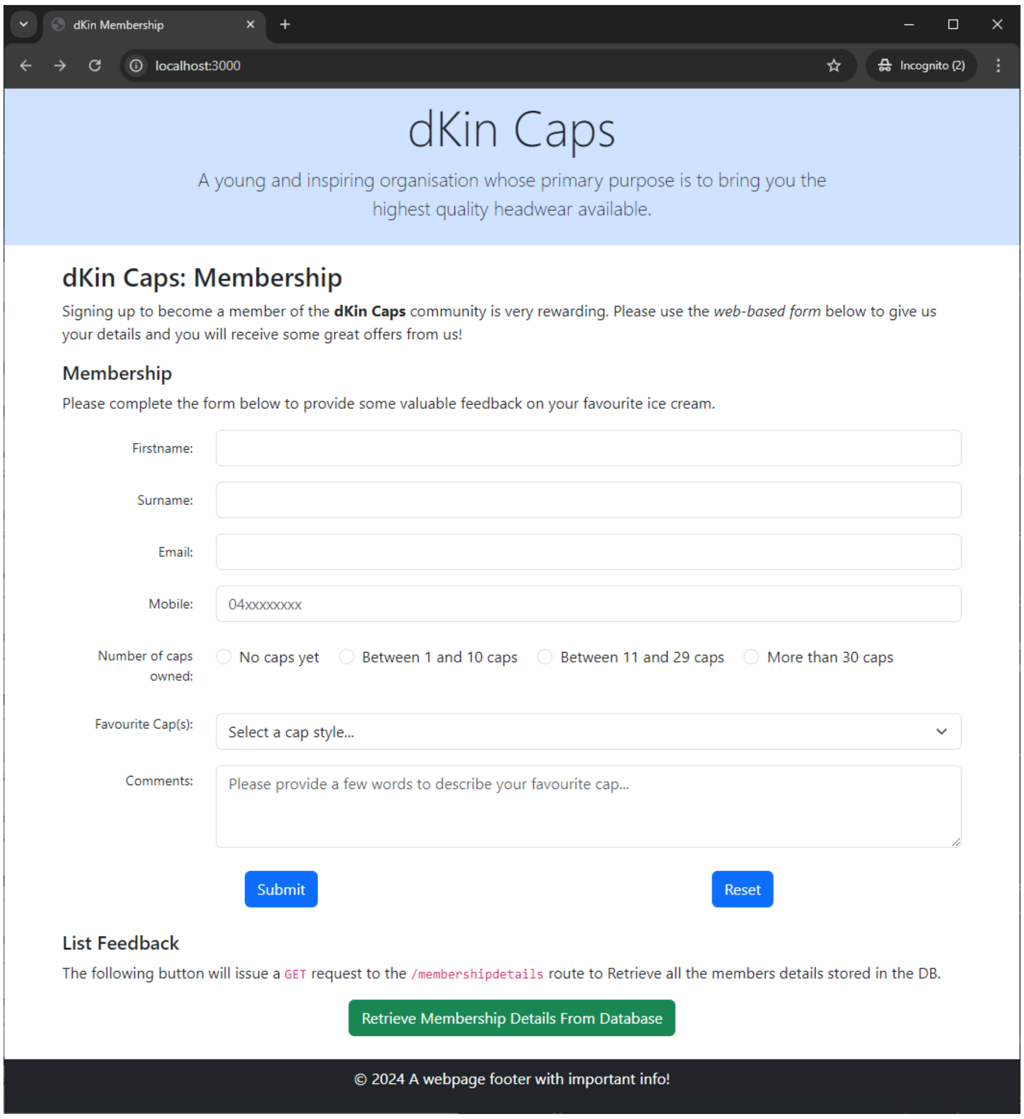
10.1P: Server Database for Website Project

Task

In this task, you are required to extend your web server from **Task 9-1P** to allow for the feedback to be stored in a DB for retrieval even if the web server has to be restarted.

In addition to the storing of the feedback, your web page should also provide a method to retrieve the contents of the database through the addition of another button. A sample screenshot of what your main page should look like is shown below:



dKin Caps

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

dKin Caps: Membership

Signing up to become a member of the **dKin Caps** community is very rewarding. Please use the *web-based form* below to give us your details and you will receive some great offers from us!

Membership

Please complete the form below to provide some valuable feedback on your favourite ice cream.

Firstname:	
Surname:	
Email:	
Mobile:	04xxxxxxxx
Number of caps owned:	○ No caps yet ○ Between 1 and 10 caps ○ Between 11 and 29 caps ○ More than 30 caps
Favourite Cap(s):	Select a cap style...
Comments:	Please provide a few words to describe your favourite cap...

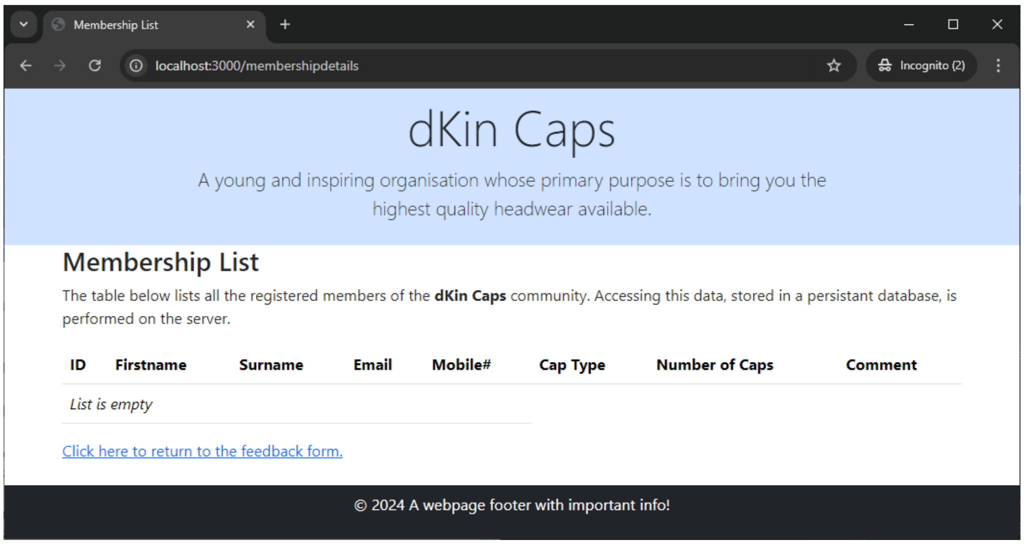
List Feedback

The following button will issue a `GET` request to the `/membershipdetails` route to Retrieve all the members details stored in the DB.

© 2024 A webpage footer with important info!

Task10.1.1 Modified form page with extra buton

Initially, the database should be empty, so clicking the *Retrieve Membership Details From Database* button should display an 'No data found' message. An example of this is shown below:



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.

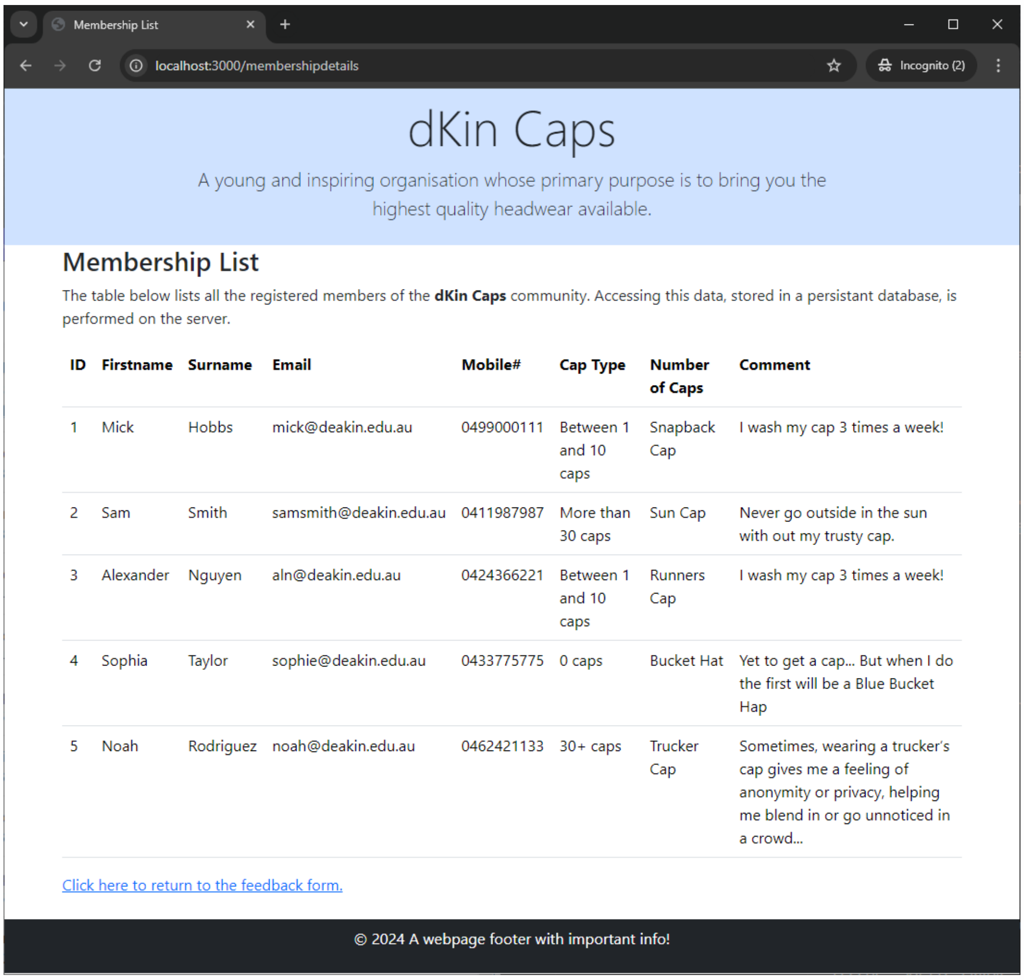
ID	Firstname	Surname	Email	Mobile#	Cap Type	Number of Caps	Comment
List is empty							

Click here to return to the feedback form.

© 2024 A webpage footer with important info!

Task10.1.2 Membership database 'empty'

When adding further membership details, these should be stored in the database, and when the *Retrieve Membership Details From Database* button is then clicked, the details should be displayed in a table:



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 3 times a week!
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...

Click here to return to the feedback form.

© 2024 A webpage footer with important info!

Task10.1.3 More form feedback completed and response page

You will need to create a web server that implements a simple *Membership Server*, with an interface to a simple database (file) and accepts requests from a user (client browser). These requests include a **POST** to store a *new* record into the database and a **GET** to display the current/updated contents of the database.

The database structure (schema) should support a record with the following fields:

Name	Type
id	INTEGER PRIMARY KEY AUTOINCREMENT
fname	TEXT
sname	TEXT
email	TEXT
mobile	TEXT
fname	TEXT
numcaps	TEXT
favourite	TEXT
comment	TEXT

> NOTE: All the fields, except for the `id` field, are of type `TEXT` (i.e., the mobile number and the number of caps are treated as strings)

Steps

To complete this task, you are required to:

1. Make a standalone *Node.js* program (i.e., a `createDB.js` file) and execute it on the *Node.js* console to create and initialise a server side *SQlite3* file database in the server folder. The database contains a table that is used to store the data sent from a form page of your own website (as described in the table above).
2. Make another *Node.js* program (i.e., a `index.js` file) in your *Node.js* server folder. This program is able to launch the server, accept the data sent from a form page of your website (in a **POST** request on the route `/submitmembership`), save the received data into the database table, and display the table data upon the request from the client (i.e., a **GET** request on the route `/membershipdetails`).

3. Make necessary changes to a form page of your website (can be stored in the template file `index.ejs`), so that it is able to:

    - send the form data to the server using a **POST** message linked to a **Submit** button
    - send a data retrieval request (e.g., a **GET** request message linked to a **Retrieve Membership Details From Database** button) to the server

4. Launch the server by executing your *Node.js* program you made.
5. Visit the form web page of your website via the local *Node.js* server (e.g., `http://localhost:3000/` which will retrieve the template view `index.ejs` page)

using a web browser.

6. Enter data into the form and submit the form to the server (three or more times, by clicking a **Submit** button). The data should be saved in the server database table.
7. Within the form page, send a data retrieval request (e.g., by clicking a **Retrieve Membership Details From Database** button) to the server. After receiving the request, the server retrieves data from the database table and displays the retrieved data in the browser.

Hints

The code snippet for the **Retrieve Membership Details From Database** button could be:

```
<h5 class="mt-4">List Feedback</h5>
<p>
    The following button will issue a <code>GET</code> request to the <code>/feedback</code>
    route to retrieve feedback stored in the DB.
</p>
<div class="d-grid gap-5d-md-flex justify-content-md-center   mb-4">
    <a href="/feedback">
        <button class="btn btn btn-success" id="getfeedbackBtn">
            Retrieve Membership Details From Database
        </button>
    </a>
</div>
```

This task is similar to the example provided in the unit site. You will note the structure of the database table will be different (slightly) and there will need to be a change to the form inputs to match the ‘fields’ expected in the database.

What will you submit?

You should submit:

- Source code of the template web page of your main page *form* (i.e., the `index.ejs` file)
- Source code of the template file that renders the contents of the database into a table (i.e., the `members.ejs` file)
- Source code of the *Node.js* file (i.e., the first `createDB.js` file) that creates a server **file** database with a table in it.
- Source code of the *Node.js* server program file (i.e., the second `index.js` file).
- Screenshot of the browser window showing the the **empty** database table.
- Screenshot of the browser window showing the form web page with entered data.
- Screenshot of the browser window showing the retrieved data from the database table after the data retrieval request is sent to the server.