8.1P: Creating a Node Project

Tasks

> This task assumes that you have already installed **Node.js** on your computer.

The goal of this task is to demonstrate the successful building and running of a **Node Express web server** that is capable of serving both **static** and **dynamic** web pages.

Your page should be able to serve the following requests:

URL	Info
http//:localhost:3000/sample.html	A static HTML file in your `public_html` folder
http//:localhost:3000/	A dynamically created page as a response
http//:localhost:3000/doesnotexist	An error response 404 - resource does not exist
http//:localhost:3000/forceerror	A route handler that induces a fault, in turn demonstrates a 500 *system* error message

Stage 1: Creating your Node Express server from the command line

You are required to **create your own local directory** (i.e., `task8_1p` and to create/install a local Node Express web server. Follow the instructions below to create your own Node project.

1. Run `npm init` in your created local directory to create the initial `package.json` file. Populate the fields with the default values (i.e., *yes* to the questions)

2. Add the two support packages needed for this project by issuing the command `npm install express morgan`:

   - *express*: The web server we have been using in the previous tasks; and
   - *morgan*: a lightweight logging package for the HTTP requests and responses processed by the web server.

3. Create a new `index.js` file in the folder containing the `package.json` file. This is where your web server code will go.

4. Edit the `index.js` file to contain the following skeleton code (note, this is similar to what was found in the `node_server_template.zip`):

```
// The package for the web server
const express = require('express');
// Additional package for logging of HTTP requests/responses
const morgan = require('morgan');

const app = express();
const port = 3000;

// Include the logging for all requests
app.use(morgan('common'));

// Tell our application to serve all the files under the `public_html` directory
app.use(express.static('public_html'));

// ********************************************
// *** Other route/request handlers go here ***
// ********************************************

// Tell our application to listen to requests at port 3000 on the localhost
app.listen(port, ()=> {
    // When the application starts, print to the console that our app is
    // running at http://localhost:3000. Print another message indicating
    // how to shut the server down.
    console.log(`Web server running at: http://localhost:${port}`);
    console.log(`Type Ctrl+C to shut down the web server`);
})
```

5. Create the folder that will hold the *static* files `public_html` .
6. Create a new *html* file in that folder called `sample.html` (**NOTE**: We do not create an `index.html` file!)

7. Edit the *scripts* section of your `package.json` file to contain the following:

```
"scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node .",
    "start:dev": "nodemon ."
},
```

8. Install the external *node monitoring package* to your system (globally) by using the command `npm install nodemon --global`. This will allow the `nodemon` to be run from your command line, as well enabling this new project script:

   - `npm run start:dev`

Stage 2: Serving pages (routes) dynamically

9. Create a handler for GET requests on the `/` route and it should return a response dynamically that contains a message and the time the request was received. Use the `Date()` object to build a string to return in the response. > **NOTE**: This handler replaces what would be the `index.html` static file.

10. Add the two additional error handlers for the following cases:

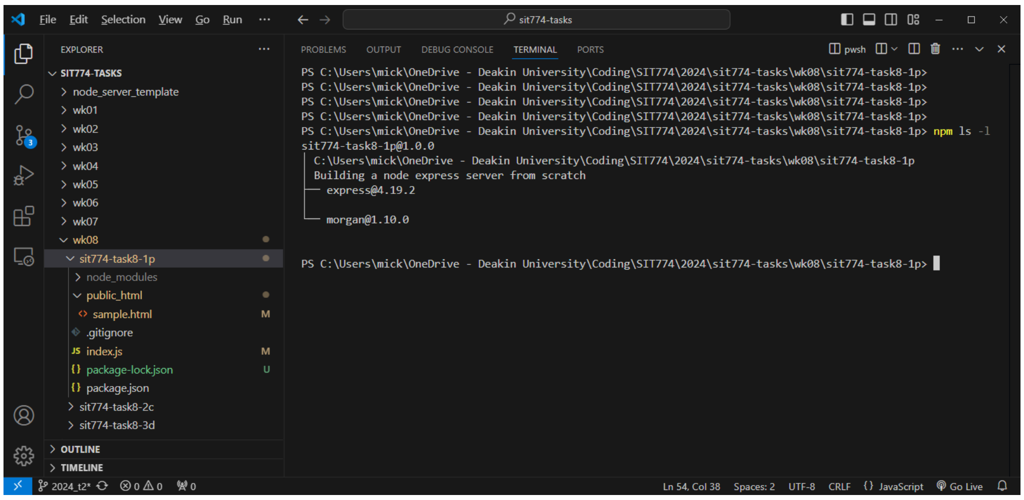
    - **404** - A generic error message to display when a file/route resource is not found
    - **500** - The generic error message for when the server code itself

experiences an error (i.e., when the JS code fails for invalid reference, etc.)

11. Add a final route handler to *cause* an error, similar to the code below:

```
// NOTE: This is not a real handler and should never be used in
// production... it is only here to demonstrate you have a valid
// 500 error handler.
app.get('/forceerror', (req,res) => {
    console.log('Got a request to force an error...');
    let f;  // empty variable
    // will cause an error as f doesn't have a method called nomethod()
    console.log(`f = ${f.nomethod()}`);
})
```

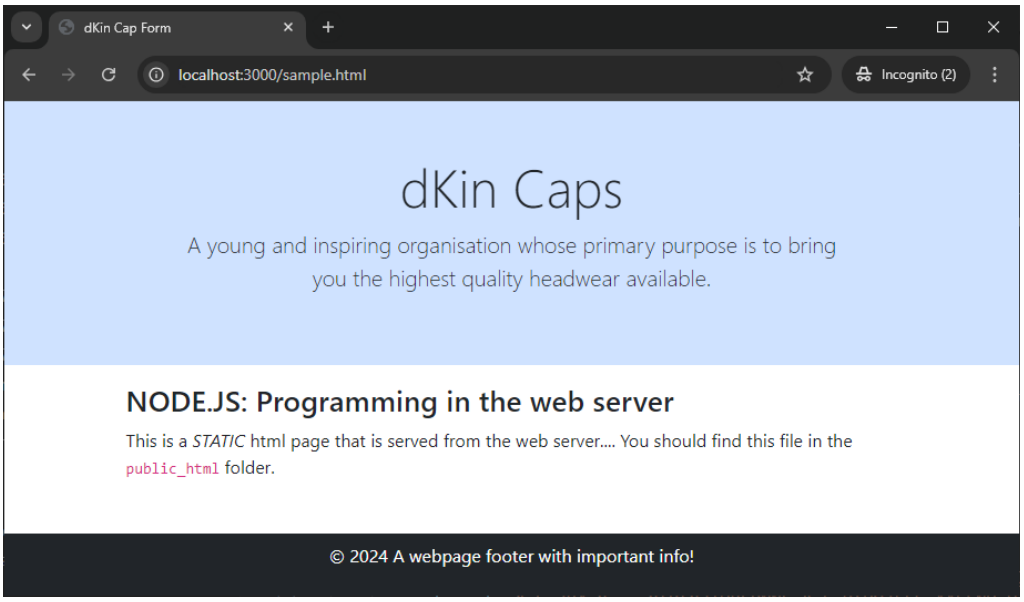
Sample Output



```
PS C:\Users\mick\OneDrive - Deakin University\Coding\SIT774\2024\sit774-tasks\wk08\sit774-task8-1p>
PS C:\Users\mick\OneDrive - Deakin University\Coding\SIT774\2024\sit774-tasks\wk08\sit774-task8-1p>
PS C:\Users\mick\OneDrive - Deakin University\Coding\SIT774\2024\sit774-tasks\wk08\sit774-task8-1p>
PS C:\Users\mick\OneDrive - Deakin University\Coding\SIT774\2024\sit774-tasks\wk08\sit774-task8-1p>
PS C:\Users\mick\OneDrive - Deakin University\Coding\SIT774\2024\sit774-tasks\wk08\sit774-task8-1p> npm ls -l
sit774-task8-1p@1.0.0
│ C:\Users\mick\OneDrive - Deakin University\Coding\SIT774\2024\sit774-tasks\wk08\sit774-task8-1p
│ Building a node express server from scratch
├── express@4.19.2
│
└── morgan@1.10.0


PS C:\Users\mick\OneDrive - Deakin University\Coding\SIT774\2024\sit774-tasks\wk08\sit774-task8-1p>
```

Task8.1.1 NPM Package Listing



dKin Caps

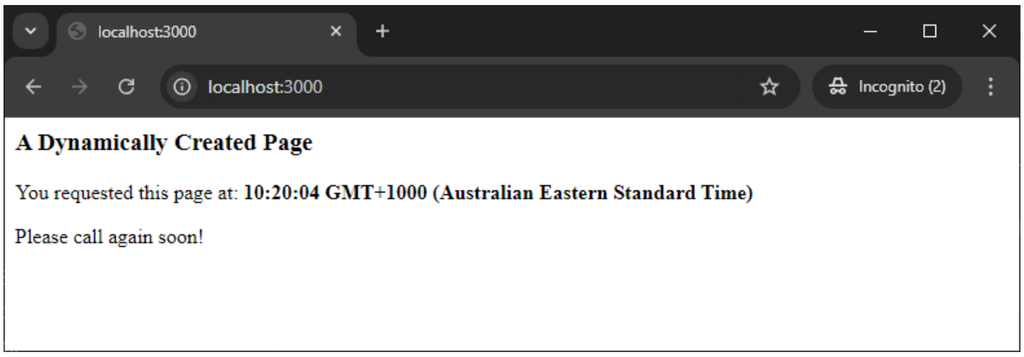
A young and inspiring organisation whose primary purpose is to bring you the highest quality headwear available.

NODE.JS: Programming in the web server

This is a *STATIC* html page that is served from the web server.... You should find this file in the `public_html` folder.

© 2024 A webpage footer with important info!

Task8.1.2 A STATIC web page

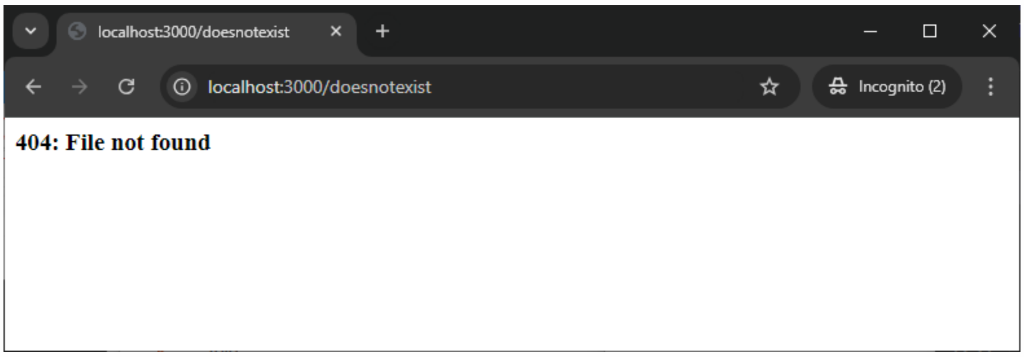

localhost:3000

A Dynamically Created Page

You requested this page at: **10:20:04 GMT+1000 (Australian Eastern Standard Time)**

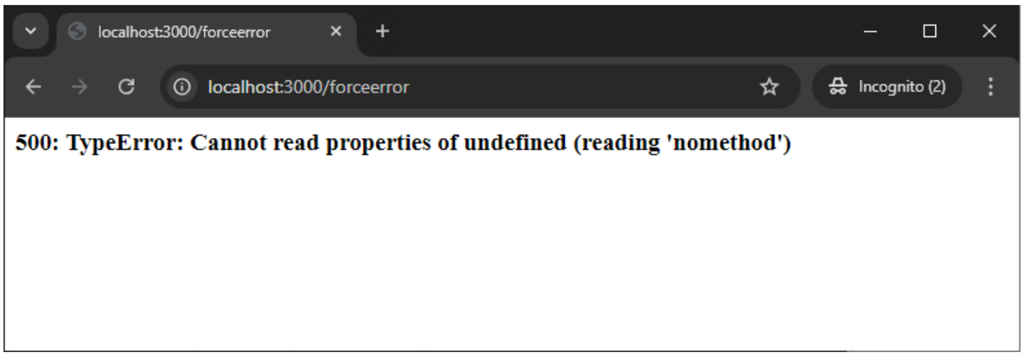
Please call again soon!

Task8.1.3 A DYNAMIC web page


localhost:3000/doesnotexist

404: File not found

Task8.1.4 A 404 Error response


localhost:3000/forceerror

500: TypeError: Cannot read properties of undefined (reading 'nomethod')

Task8.1.5 A 500 (5xx) Error response

What will you submit?

You should submit:

- Source code of the file `index.js`.
- Screenshot #1 - your command line use `npm ls -l` from your project folder to obtain a listing of the installed packages, as shown in the Image 8.1.1 above.
- Screenshot #2 - the browser window showing the result of visiting a **Dynamically** generated root page, e.g., `http://localhost:3000/`.
- Screenshot #3 - the browser window showing the result of visiting your **Static** HTML file, e.g., `http://localhost:3000/sample.html`.
- Screenshot #4 - the of the browser window when a file/route/resource is not found, e.g., `http://localhost:3000/doesnotexist.html`.
- Screenshot #5 - the browser window when fault generating route is called, e.g., `http://localhost:3000/forceerror`.