F28WP Web Programming
Lab report
Lab 3

Student full name: Akshay Arunkumar Garg

Student HWU username: ag2006

Student's GitHub URL of the Lab: https://github.com/ag220502/Lab3NodeJs

Demonstrated to Lab helper: .....

Mode of demonstration: ☐ face-to-face ☐ online

Date of demonstration:30th November 2021

Time of demonstration: .....

The report should be about 3 pages long (excluding the title page) and submitted in PDF format. All explanations must be brief and supported by short code statements, and possibly illustrated by precise screenshots.

1- Create web server (2 marks)

Explain the structure of package.json

This file includes details such as name, version, and description of the project. The main property in this file gives direction from where to start executing the module which is the entry point. For our project it is app.js. The scripts property works like a dictionary which is key-value pair and each of these keys include the command which we can execute and the values of these keys are the command which will be executed when we will run the command using the keys.

Explain how to use npm to install packages.

The npm install command is used for downloading packages in our project. This also downloads the dependencies in the package.json file and this command can be used with or without arguments. When we run without arguments, it creates a node_modules folder where node modules are installed. Else, when we run with arguments it installs the specific module to the node_modules folder.

What are express and nodemon modules?

The express mode in Node js is used to create the web server by writing the command npm install express in the specific folder and the content of these modules will be available there in the node_modules folder in the current directory. Whereas nodemon is a tool that is used to run the code without writing commands again and again. Here, whenever we make changes, it automatically restarts the server so that we have don't have to write the command again and again.

Explain how the HTTP server is created in your app.js

In the app.js file, we create an app and then call the express module in it. Then, we create a port and specify it as 3000. After this is done, we create the server and make it listen on the port which we created which is 3000, or on the environmental variable.

2- Static web pages (1 mark)

Explain why do we need the static middleware?

We need the static module to send all the static files present in our project. For example files in the public folder are static and we are using this module to send all the files like the HTML, CSS files or the images, js files.

How does it work? Show example from the code.

Code

```
app.use(express.static ('public'));
```

Explanation

In this code, we are sending the static files using the module which are under the pubic folder. Such as index.html, hwu.css, etc.

3- Using ejs (1 mark)

Explain briefly ejs. Explain the structure of the ejs files (header, footer, index) and their dependence.

Ejs is also known as embedded javascript which is a tool used to create web pages using javascript that includes dynamic content. The header file contains the code to display the header on the web page which is there on all the pages. It helps us to reduce the code so that we don't have to write the code again and again. In the footer, we are doing the same but in this footer file code of the footer is there. In the index file, we are calling bot the files and then some code of content is also present.

Show code excerpts from app.js that allow setting ejs and sending the static HTML content from index to the browser.

```
app.set('view engine', 'ejs');
app.get("/", (req, res) => {
    res.render('index'); //no need for ejs extension
});
```

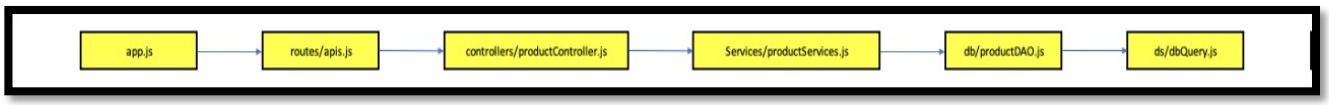
Here, we are setting the view engine as ejs and then we are loading the index file as the default file.

Explain how your contacts.ejs works.

In this, we are including the content of the header and the footer file dynamically and then e is creating a route for the contacts page using the get method. This method gets the specified URL from the network and puts the contacts of the page to the URL.

4- Displaying the catalogue (2 marks)

Draw a simple diagram to explain the dependence of the following files



Briefly explain this structure and the role of each file (what is it responsible for?) Illustrate with code excepts.

- **app.js**
  The project starts executing from this file which acts like an entry point.
- **routes/apis.js**
  In this file all the routes are defined and also, it includes the url's and calls the method in productController.js file.
- **controllers/productController.js**
  In this file, we are calling the method to extract the data which is returned when query is executed and send them to specific pages.
- **services/productServices.js**
  In this file we are calling the method from productDAO.js and checking if the query executed returns some result or not.
- **db/productDAO.js**
  This is the file where queries are written, and we are calling the methods from dbQuery file to check whether the query is executed or not after connection is established.
- **db/dbQuery.js**
  This file is used to check whether the connection with the database is established so that queries can be executed. Also, this file takes care of the query as well that whether the query is executed or not.

What is the text displayed on the browser when the "catalog" request is successful?

When catalog request is successful the data of articles table is displayed on the web page.

Selected articles

Name	Reference	Category	Price
High chair	05-303	Furniture	1090AED
Round table	07-390	Furniture	1820AED
Midimu-hight cabinet	08-151	Furniture	3995AED
Juxtaposable low cabinet	09-077	Furniture	1430AED
Miniter desk	11-518	Furniture	3190AED
Half ministr desk	11-543	Furniture	2010AED
Secretary desk	11-573	Furniture	2890AED
Dactylo desk	11-583	Furniture	1890AED
Mobile 3-drawers cabinet	14-040	Furniture	2410AED
Printer stand	15-991	Furniture	1370AED
Archive box green (hundred)	25-069	Stationay	344AED
Archive box red (hundred)	25-070	Stationay	344AED
Archive box blue (hundred)	25-071	Stationay	344AED
Archive box yellow (hundred)	25-072	Stationay	344AED
Archive box black (hundred)	25-073	Stationay	344AED
Archive box grey (hundred)	25-074	Stationay	344AED
Archive box marron (hundred)	25-075	Stationay	344AED
USB stick, 2 GB	31-068	Digital	8.8AED
USB stick, 4 GB	31-069	Digital	15.8AED
USB stick, 16 GB	31-070	Digital	6.6AED
USB stick, 32 GB	31-071	Digital	12.4AED

5- Ejs for dynamic content (2 marks)

Explain how catalogue.ejs works.

Using code excerpts (the function calls), show the flow of execution starting the user click on Catalog (in the web page) and ending with the actual table of products showing on the web pages.
Explain the different steps taken to generate this HTML that displays the table of products. Mainly explain how the data is sent from the controller to the ejs file.

- When the user clicks on 'Catalog', they will be taken to /api/catalog where a callback is made to the getCatalogue method of productController.js.

```
router.get('/api/catalog', productController.getCatalogue);
```

- The getCatalogue method uses the searchService method of productServices.js and is used to send the result to the catalogue file.

```
<% products.forEach(product => { %>
    <tr>
        <th scope="row">
            <%= product.name %>
        </th>
        <td>
            <a href="/api/article/<%= product.reference %>">
                <%= product.reference %>
            </a>
        </td>
        <td>
            <%= product.category %>
        </td>
        <td>
            <%= product.price %>AED
        </td>
    </tr>
<% }) %>
```

- The searchService method uses the findAll method of productDAO.js and is used to see if it receives any result.
- The findAll method then uses the getResult method of dbQuery.js to check if the sql query written is being executed without any errors and checks the database connection as well.
- Finally, if the callback functions return without any errors, then a table displaying all the products will be displayed.

Explain how you implemented the request for displaying a single article. In particularly, explain how you created article.ejs.

- Initially we create an article.ejs page. When the user clicks on a specific article, they will be directed to the url '/api/article/:id' where a callback is made to the getProductByID method of productController.js.
- The getProductByID method then uses the searchIDService method of productServices.js and is used to send the result to the article file.
- The searchIDService method in turn uses the findByID method of productDAO.js and is used to see if it receives any result.
- The findByID method then uses the getResult method of dbQuery.js to check if the sql query written is being executed without any errors and checks database connection as well.
- Finally, if the callback functions return without any errors, then the article page along with the details of the selected article will be displayed.

6- Login (1 marks)
Explain your implementation of the login feature. List all files required for this functionality and explain their role.

Explain how the parameters (login and password) are sent by the client and processed by the server.
Explain the password processing in the server side.
Show how you used ejs to send the login result to the client.
First we create a login page which takes username and password as input from the user. When submitting the form the user is redirected to the /api/login from where a call to the loginControl function of the clientController.js is made.
The loginControl method gets the values using the name attribute which then calls the loginService function and renders the response to the postLogin.ejs page(which is like a welcome page after the sign in for the user).
To verify whether the user is already a user or not the loginService method calls the findByUsername method. If the user is found then it checks for the password using method in the clientDAO.js file. the clientDAO.js has methods containing the queries to check for the username and password checking.The cryptPassword function used for hashing the password is also there and is used when a new user is created. After this it calls the getResult method to execute the query.

Part 7: **Additional features (1marks)**
You need to describe only the features you implemented. Implementing correctly one feature is sufficient to get 1 mark).

- **Listing clients**: *explain how you implemented the list of clients and how yopu protected this functionality to make it accessible only to logged user with admin privilege. This requires checking that the user is logged in and has admin privilege before allowing the execution of this use case. To. Do this, you need to use the session and set the username in the session.*
- ***Display a single client**: Explain the flow of execution and the files you added to implement the feature that allows displaying the details (all attributes) of a single client giving their id.*
- **Add product to cart**: *explain your implementation of the cart functionality. The cart is a list of products and quantities. You should explain how you set up the cart into the session.*
- **Cart management**: *explain your implementation of the cart management: modifying the quantity for a given product in the cart, deleting a product from the cart, clearing the cart.*
- **Cart display**: *explain how you calculate the price including VAT of each product and how you calculate the total price of the cart (sum of the prices of each product). Show your display of the cart with the prices, VAT and total.*
- **User checkout**: show the changes you did on the database to store the cart. Explain how you implemented the insertion of the cart into the database.