Full Stack Development with AI
Lab 10.2 – Creating a Complete Web Application with Express.js

Lab Overview

In this lab, you will learn how to create a server-side rendering web application using Express.js. The web application will simulate the administrator panel of a typical business-to-consumer (B2C) ecommerce shopping website. The use cases will allow an administrator to perform basic data manipulation operations on the product catalogue, i.e., CRUD or create, retrieve, update and delete.

You will be using the Pug template engine and persist product data into an SQLite database.

The synthetic product data consists of 50 products over 10 categories.

The resources that you need for this lab can be found in "resources.zip".

Exercise 1 – Creating a new Express.js Application

1. Use the file explorer or terminal on your computer to create a new folder `app`.

   A typical location to create the new `app` folder is your home directory. On a Windows computer, this will be `C:\Users\<username>`. On a macOS computer, this will be `/Users/<username>`.
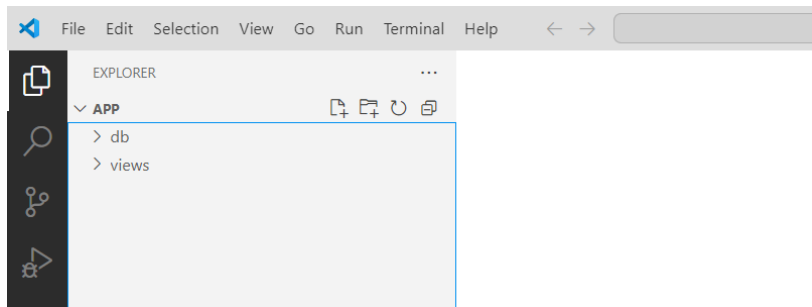
2. Start Visual Studio Code (VS Code) and open the newly created `app` folder. This folder will be empty as shown in the figure below:
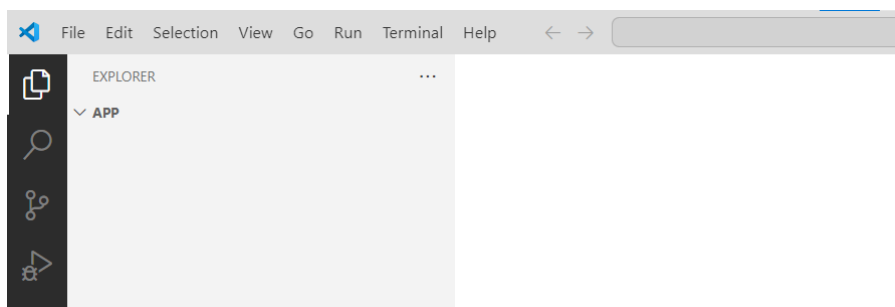
3. Right click on the Explorer pane and select "New Folder" to create a new folder `db`. Repeat the same steps to create another new folder `views`.
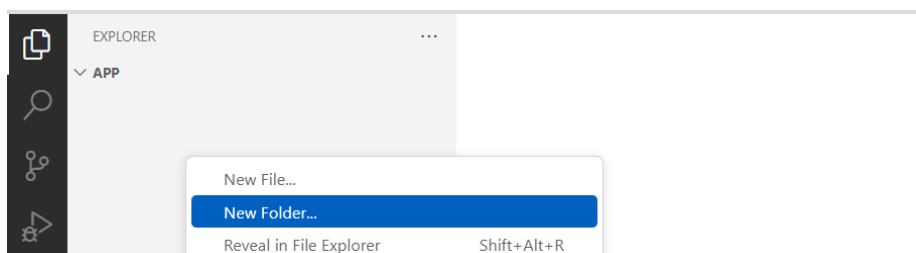
4. Your Explorer should now resemble the figure below:


5. Start a terminal in VS Code.

6. Run the following commands to install the required node packages using `npm`:

```
npm install express
npm install pug
npm install sqlite3
```

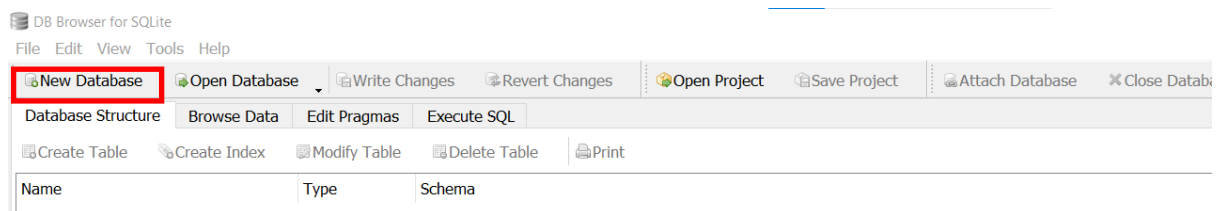## Exercise 2 – Creating a New SQLite Database

Follow the instructions below to install the SQLite Viewer extension in VS Code.

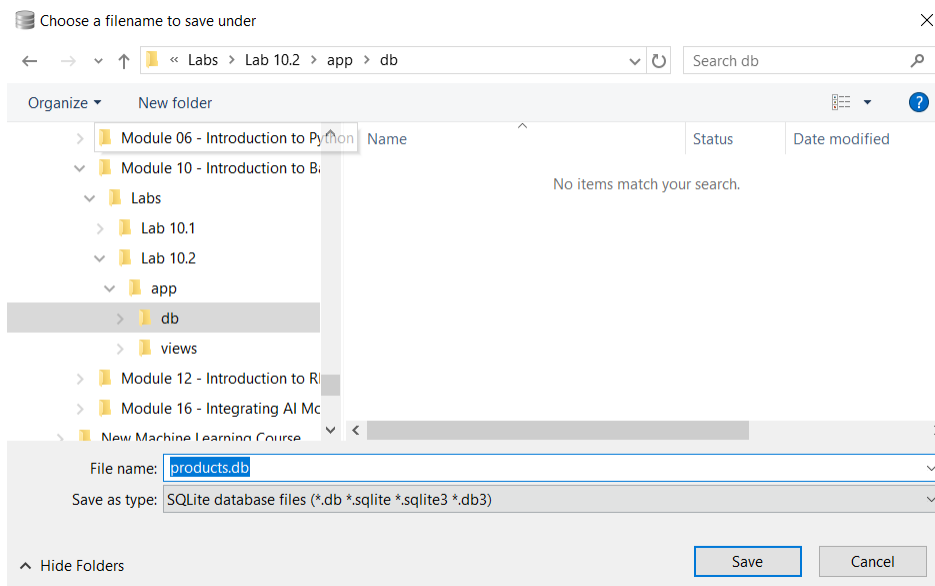1. Go to the Extensions pane and search for "sqlite". Then select "SQLite Viewer" and install it.
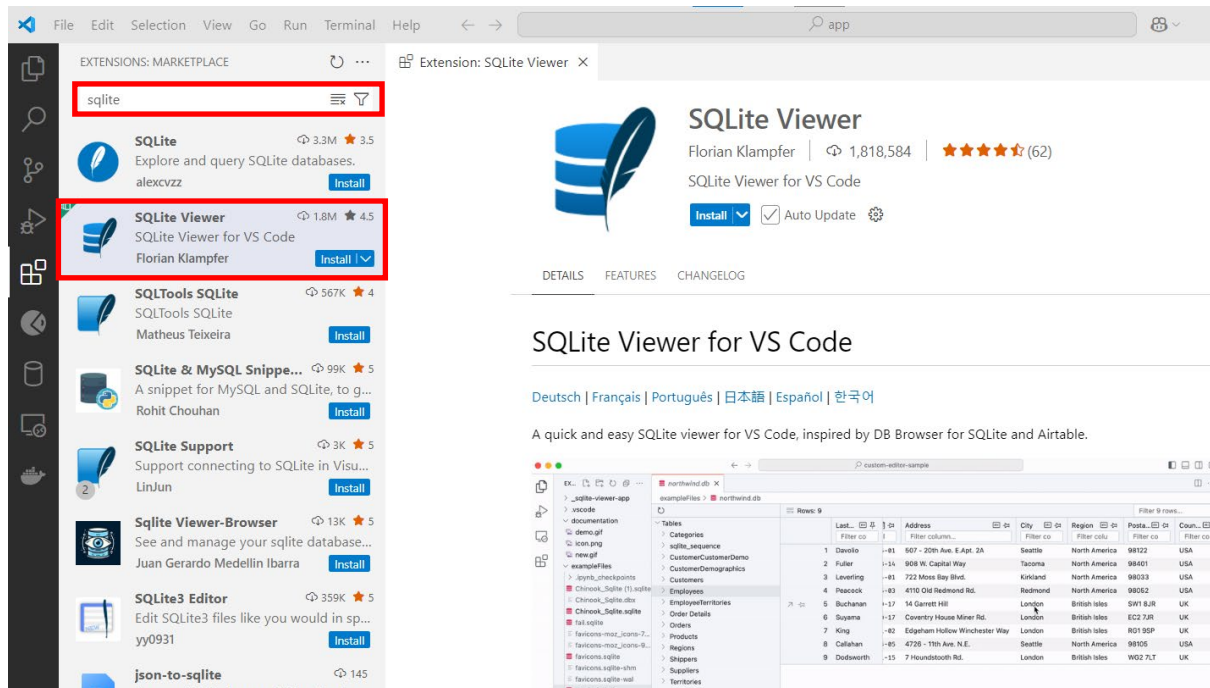
Next, follow the instructions below to create a new SQLite database:

1. Start DB Browser for SQLite.

2. Click on "New Database"

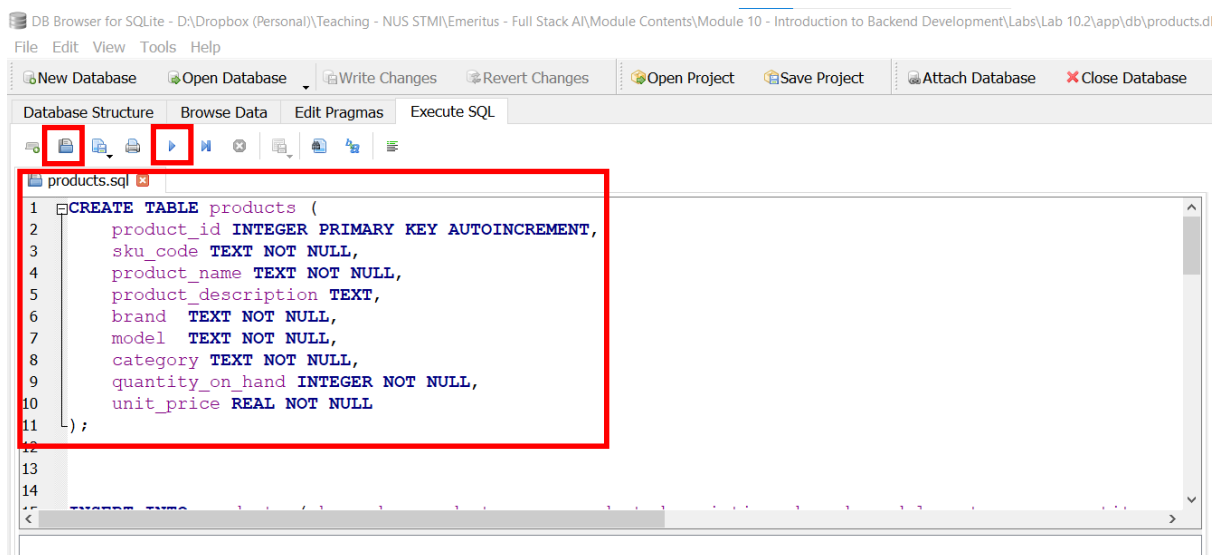3. Navigate to the app\db folder that you have created earlier.

4. Name the new database as "products.db" and click "Save".

5. Close the "Edit table definition" dialog box.

6. Switch to the "Execute SQL" tab.

7. Click on the "Open SQL File" button and navigate to the folder where you have extracted "resources.zip". Then select "products.sql" and click on the "Run" button.

8. After the SQL statements have been executed, switch to the "Database Structure" and "Browse Data" tabs to inspect the "products" table and data.

9. Click on the "Write Changes" button and then "Close Database". This will save the changes to "products.db".

10. Return back to VS Code and look under the "app\db" folder. Ensure that you can see "products.db".

11. Select "products.db" and you should be able to view the "products" table and its data within VS Studio using the SQLite Viewer extension.
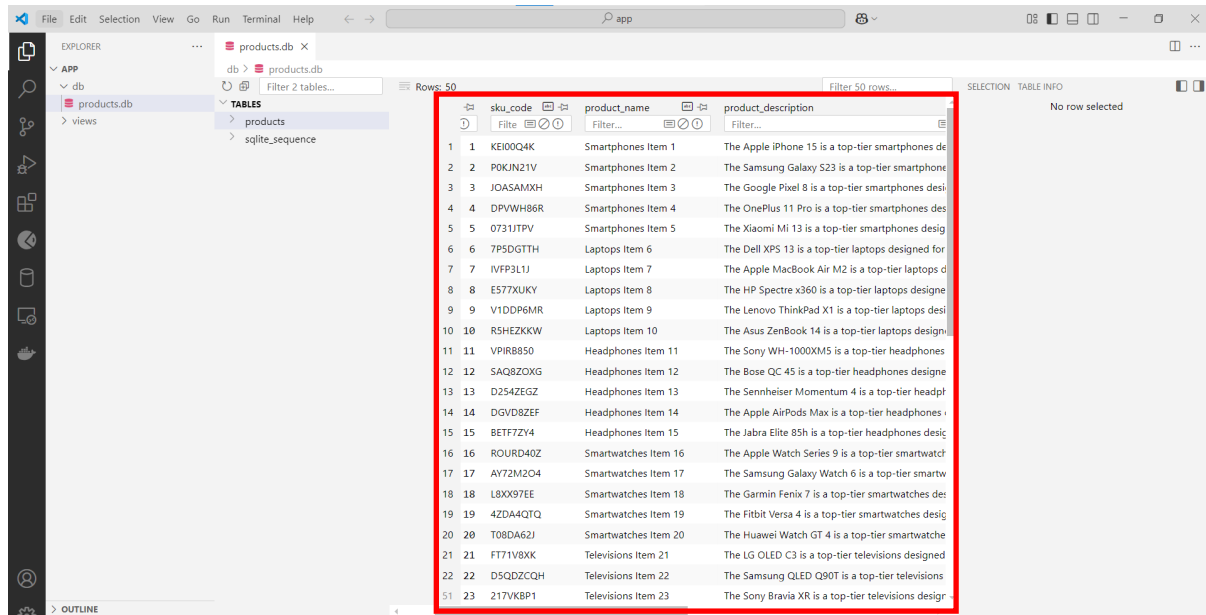
12. Take note that the SQLite Viewer extension does not support query running. If you wish to run any SQL statements, you will need to use DB Browser.

Exercise 3 – Creating New Endpoints

In this exercise, you will be creating a series of endpoints using the GET and POST HTTP methods to implement the create, retrieve, update and delete use cases for product records.

1. Right click on the Explorer in the area outside of any subfolder and select "New File". This will create the new file in the root of the web application or the app folder itself. Name the file as "app.js"

2. Paste the following code fragment representing the basic structure of an Express.js application into "app.js"

```js
const express = require('express');
const app = express();
const port = 3000;

app.set('view engine', 'pug');

app.use(express.urlencoded({ extended: true }));

app.listen(port, function () {
    console.log(`Express app listening on port ${port}!`);
});
```

3. Your VS Code should resemble the following figure:

```
JS app.js  ●
JS app.js > ...
  1   const express = require('express');
  2   const app = express();
  3   const port = 3000;
  4
  5   app.set('view engine', 'pug');
  6
  7   app.use(express.urlencoded({ extended: true }));
  8
  9   app.listen(port, function () {
 10       console.log(`Express app listening on port ${port}!`);
 11   });
 12
```

4. Between line 5 and 7, add a GET endpoint to the root path / that renders the pug template "index.pug" as response.

5. Add a GET endpoint to the path /viewAllProducts that performs the following two tasks:

   a. Retrieves all the product records from the products table using a SELECT query.
   b. Renders the pug template "viewAllProducts.pug" as response

6. Add a GET endpoint to the path /createNewProduct that renders the pug template "createNewProduct.pug" as response.

   Note that this template will render a blank HTML form for user to input the data for a new product record.

7. Add a POST endpoint to the path /createNewProduct that performs the following two tasks:

   a. Retrieves the form data in the pug template "createNewProduct.pug".
   b. Insert the data for the new product record into the products table using an INSERT statement.
   c. Return an information message together with a link back to the homepage.
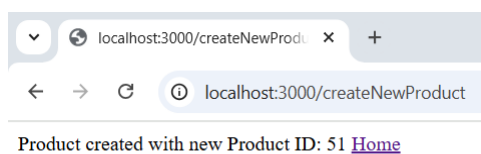
8. Add a GET endpoint to the path /updateProduct that performs the following two tasks:

   a. Accepts a path parameter for the product id of the product to be updated
   b. Renders the pug template "updateProduct.pug" as response.

   Note that this template will render a HTML form prepopulated with the existing data of the selected product for user to edit.

9. Add a POST endpoint to the path /updateProduct that performs the following four tasks:

   a. Accepts a path parameter for the product id of the product to be updated
   b. Retrieves the form data in the pug template "updateNewProduct.pug".
   c. Update the data for the existing product record into the products table using an UPDATE statement.
   d. Return an information message together with a link back to the homepage.
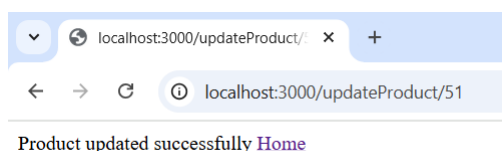
10. Add a GET endpoint to the path /deleteProduct that performs the following two tasks:

   a. Accepts a path parameter for the product id of the product to be deleted
   b. Renders the pug template "deleteProduct.pug" as response.

   Note that this template will render a table showing the existing data of the selected product for user to confirm the deletion.

11. Add a `POST` endpoint to the path `/deleteProduct` that performs the following three tasks:

    a.   Accepts a path parameter for the product id of the product to be deleted

    b.   Delete the existing product record from the `products` table using a `DELETE` statement.

    c.   Return an information message together with a link back to the homepage.
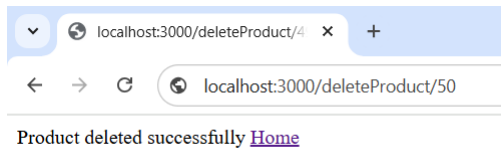
Exercise 4 – Creating the Views as pug Templates

In this exercise, you will be creating a series of pug templates representing the views of the web application. Recall that the pug code will be converted to HTML on the server side before being returned to the web browser.

Index or Homepage of the Web Application

1. Right click on the `views` folder and select "New File". Name the file as "index.pug"
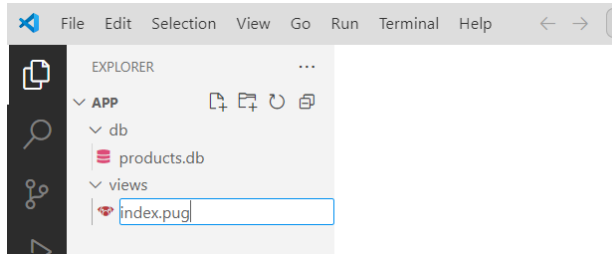
2. The new file will be opened for editing.

3. Input the pug code to create a web page resembling the figure below:
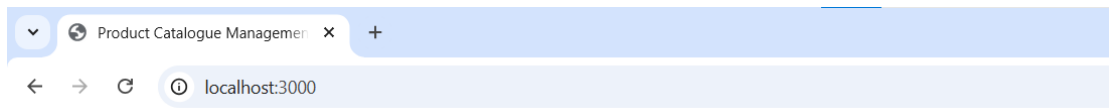
The two links should link to the following paths:
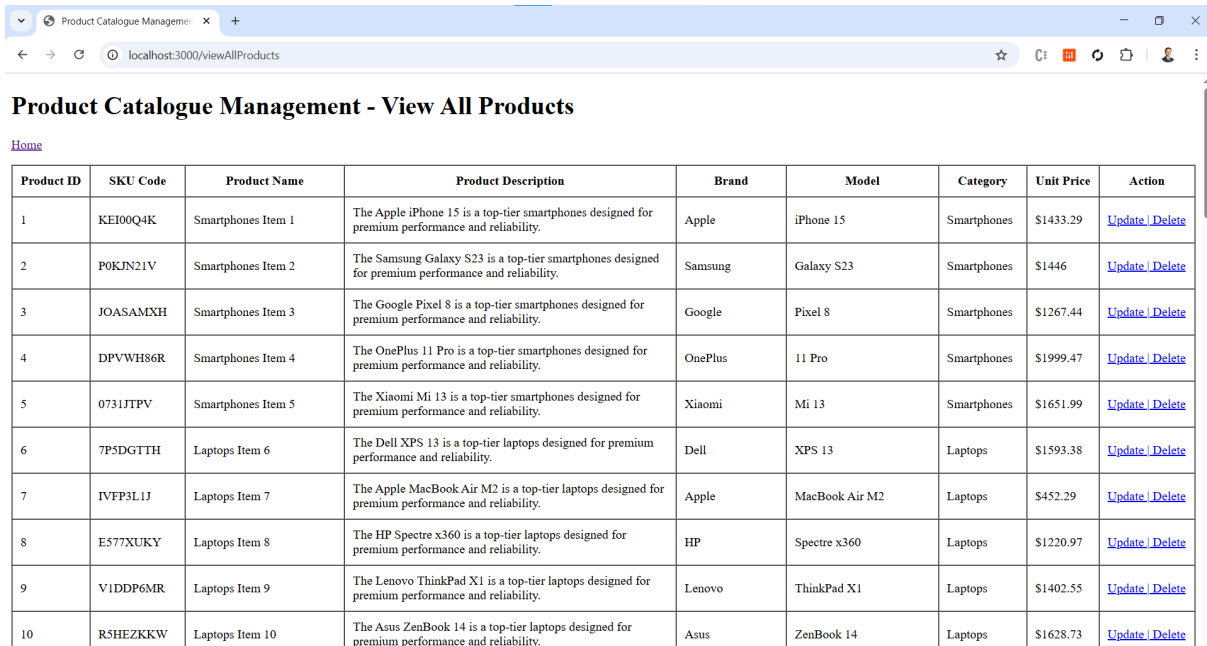
- View All Products – /viewAllProducts
- Create New Product – /createNewProduct

View All Products

1. Right click on the `views` folder and select "New File". Name the file as "viewAllProducts.pug"

2. Input the pug code to create a web page resembling the figure below:

The two links should link to the following paths:

- Update – /updateProduct/:product_id
- Delete – /deleteProduct/:product_id

Create New Product

1. Create a new pug template file "createNewProduct.pug"

2. Input the pug code to create a web page resembling the figure below:

Update Product

1. Create a new pug template file "updateProduct.pug"

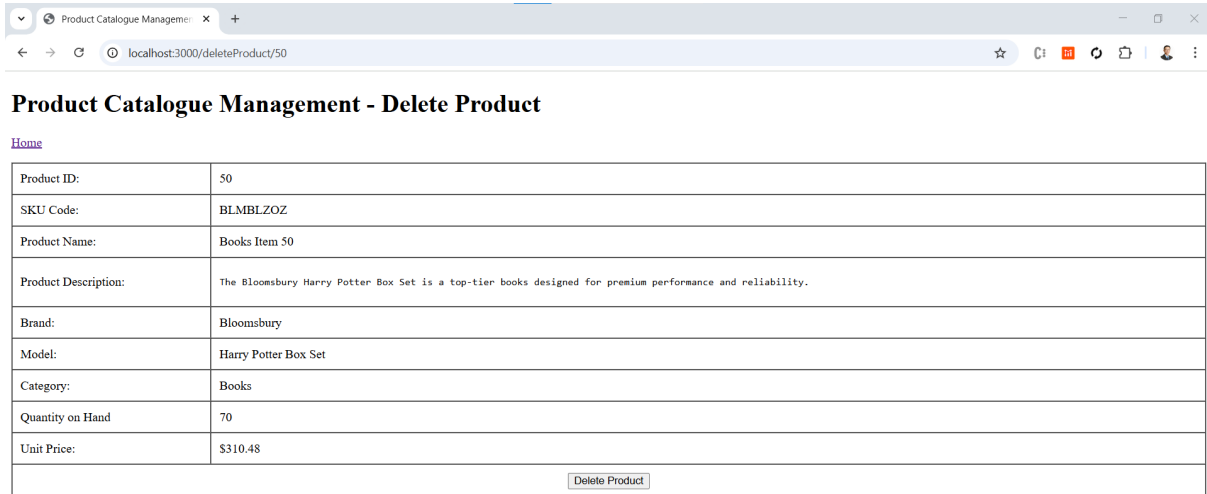2. Input the pug code to create a web page resembling the figure below:

Delete Product

1. Create a new pug template file "deleteProduct.pug"

2. Input the pug code to create a web page resembling the figure below:

Exercise 5 – Running and Testing the Web Application

Follow the instructions below to run the web application:

1. Start a new terminal in VS Code.

2. Run the following command to run the web application:

```
node app.js
```

3. Open your web browser and navigate to the address http://localhost:3000/. Ensure that you are able to view the index or homepage.
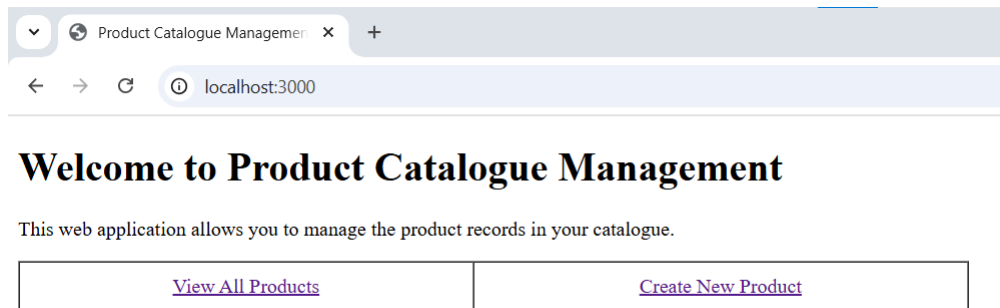
4. Click on the "View All Products" link to view the product records and perform some update and delete operations.

5. Click on the "Create New Product" link to add a new product record. Then return to the view all products webpage and ensure that you can view the new product record.

-- End of Lab --