Assignment 3
Out of 50 Marks

DUE: 05 June 2024

IMPORTANT NOTES:

- This is an individual assignment.
- Homework assignments are based on assessment objectives. If an objective has been achieved, a mark will be allocated.
- **All assignments are submitted via ClickUP. See the Assignments section**.
- **You only upload your Angular App (.zip), API (.zip), and video demo (.mp4).**
- **You may use any of the previous source code shared during the lectures to assist in completing this Assignment. See the code on ClickUP under Course Content. You can then build upon it**.
- **Please execute the API migration before building your angular application**. **See the Angular and API installation and configuration section.**
- **If you are caught plagiarising, we will give you zero percent (0%), and you will be reported for plagiarism immediately.** We will audit historical assignments throughout the semester. **We trust that you understand the importance of this point.**

VIDEO INSTRUCTIONS:

- Make sure that everything is running when you start recording the video. The video should not be longer than 15 minutes showing the items in the **Standard Requirements** against the **Rubric**.
- When showing something from the **Standard Requirements**, show us as much detail as required. ***See the Rubric for the assessment criteria***. For example, when assessing the "***Program Functionality,***" you must show the validation working per page, page redirects/navigation, the file upload works, the data is saved to the database, the password is hashed, and the pages are working as expected. Similarly, for the "***Program Output,***" the correct notification messages are displayed for the relevant pages, the side-menu displays correctly depending on whether the User is logged in or out, the Product Listing page shows the correct data in the valid format, etc., the active products and charts are pulled through to the Product reporting page, and all the pages are demonstrated. Further, for the "***Code readability***" we expect you to show us your code and display the organization of the code and descriptive names (*i.e., all the code used to create the program,* ***not the configuration files like package.json****, etc.*). **The same applies to the rest of the Rubric. See below.**
- If something did not work in your code, in the video, explain to us what you wanted to do and what you wanted to achieve with your approach. **This is to assess you correctly according to the Rubric**.
- **See the "Video Recording and Compression Guide" in the Assignments section on ClickUP for video recording and compression assistance.**

SUBMISSION INSTRUCTIONS:

- In this Assignment, you will be given the requirements to implement.
- **Source Code:** Zip your source code files together, and for the API name it **uXXXXXXXX_HW03_API.zip**, where the XXXXXXXX is your student number, e.g., u12345678_HW03_API.zip. Further, for the Angular App, name it **uXXXXXXXX_HW03_Angular.zip**, where the XXXXXXXX is your student number, e.g., u12345678_HW03_Angular.zip.
- **Video Demo:** **Do not** zip your **video demo**. In other words, submit the actual "**.mp4**" file. Name the video demo **uXXXXXXXX_HW03.mp4**, where the XXXXXXXX is your student number, e.g., u12345678_HW03.mp4.
- If files are uploaded to the wrong upload area, we will not look for the upload. Uploads should be submitted correctly.
- **Please Note**: **If you omit the code (.zip) or the video (.mp4) submission, you will lose 50% of your assignment mark**. **If no files are uploaded (neither the .zip nor .mp4), you lose 100%. Please take this seriously and plan accordingly to submit it on time**.

- ***Note: you upload the code (.zip files) and the video demo (.mp4 file) together in the same location in the Assignment 03 Submission section. See the ClickUP information in the Assignments section (when readily available).***
- Please **do not** upload the "***node_modules"*** and "***.angular"*** folders for the **Angular App**. In other words, once you have completed your program and created your video, delete the "***node_modules"*** and "***.angular"*** folders. As the Lecturing Team, we will reinstall the ***node_modules*** folder dependencies using the "***npm install***" terminal command, *where necessary*. **This is so you do not take long to upload your code with the video demo.**
- In addition, **do not** upload the "***bin"*** and "***obj"*** folders for the **.Net API application**. In other words, once you have completed your program and created your video, delete the "***bin"*** and "***obj"*** folders.

SUBMISSION DEADLINE: 05 June 2024

- There shall be no extensions to the deadline above.
- If homework submissions are uploaded too late, then upload errors **will** happen.
- Do not wait until the last minute to complete the Assignment.
- Start working on the Assignment as soon as possible.
- Email submissions **will not** be accepted.
- Late submissions **will not** be accepted.
- **No exceptions will be made for anyone.**

USE CASE:

- A client requests you to create a proof-of-concept application using Angular and .Net 6 Web API. They want to register, log in, record, peruse inventory products, and produce charts and report on active products.
- You are requested to develop the back end using a **.Net 6 API** and the front end using **Angular**.
- For the application, you need to build the capability for users to Create, Read and Report on products stored in the SQL Server 2022 database and implement Authentication functionality via login and registration.
- When the application is launched, the landing page must be the **Login page**, and navigation to all other pages must be done via angular routing, subject to the restrictions that will be detailed under "***Standard Requirements***".

STANDARD REQUIREMENTS:

- Login page:
  - The login page requires a **Username** (**an email address**) and a **Password** to log in. Logging in must be prevented if the Username or Password is not provided (Fig. 1).
  - When the User clicks on the link to register by "***Don't have an account? Register here***" they must be redirected to the Register page (*see the Register page section*).
  - When the User enters valid user credentials, the User must be redirected to the Product Listing page (*see the Product Listing page section*).
  - After logging in, a side menu bar with Product Listing, Add Product, Product Reporting and Logout should be navigational with the relevant functionality (Fig. 2).
  - Clicking on the Logout menu item should Logout the User and return them to the Login page, hiding the side menu bar (Fig. 3).

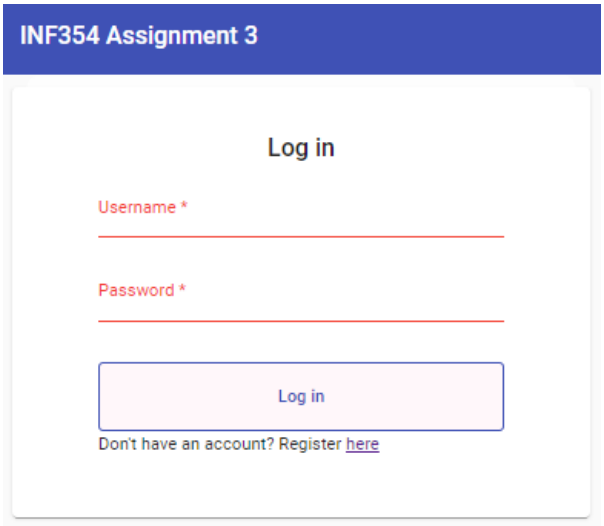


Fig. 1

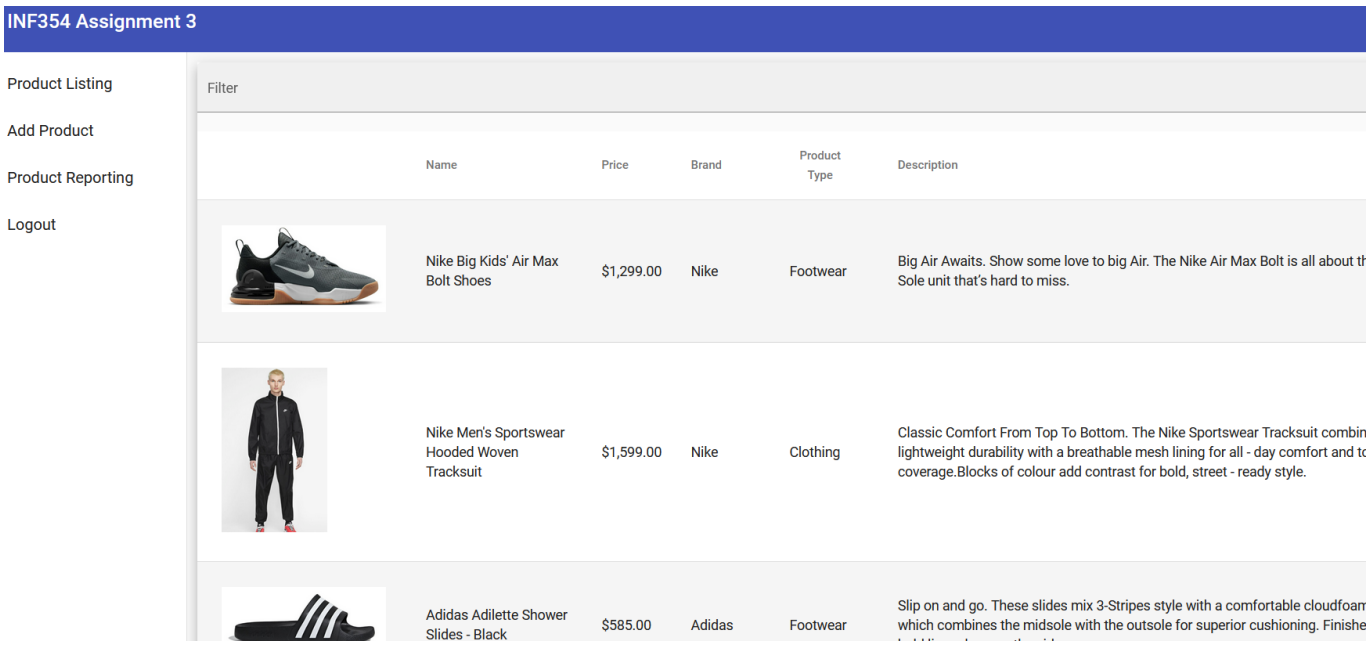


Fig. 2

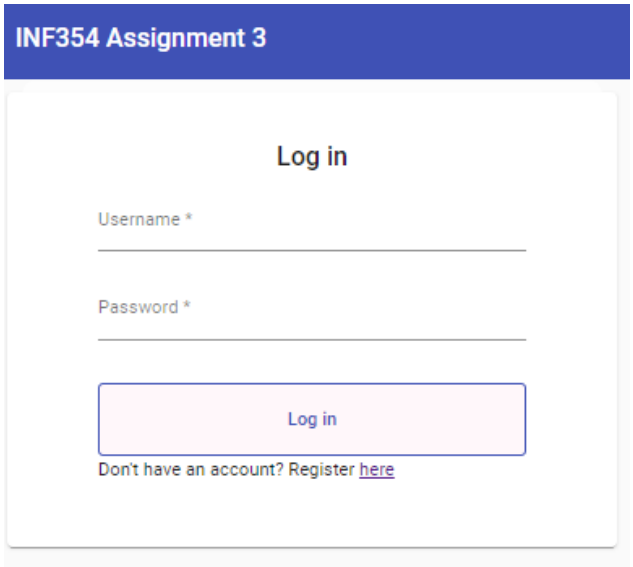


Fig. 3

- Register page:
  - The register page requires an **email address** and **password** (Fig. 4).
  - When the User is successfully registered, they must be redirected to the Login page with the following notification message "**Registered successfully.**". (Fig. 5)
  - Note: The password stored in the database must be hashed.

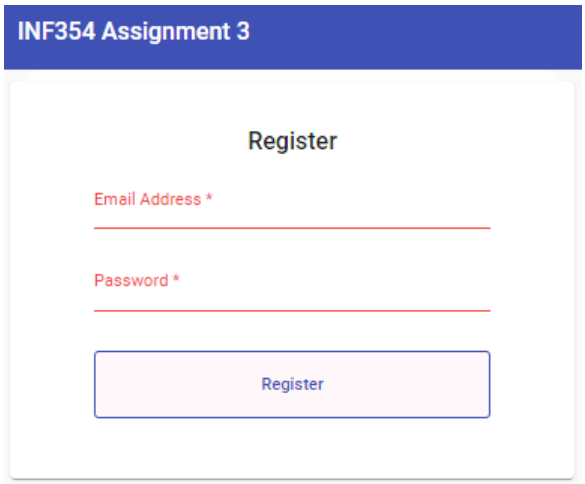
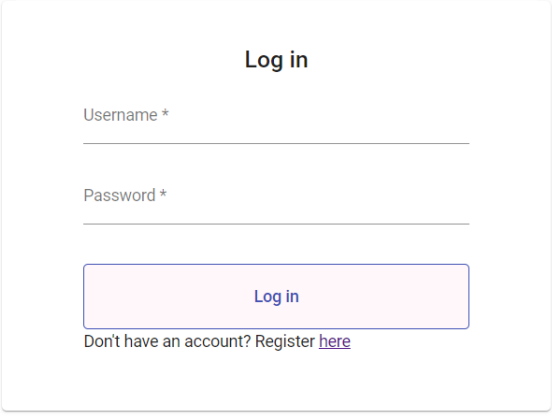


Fig. 4



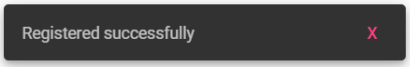
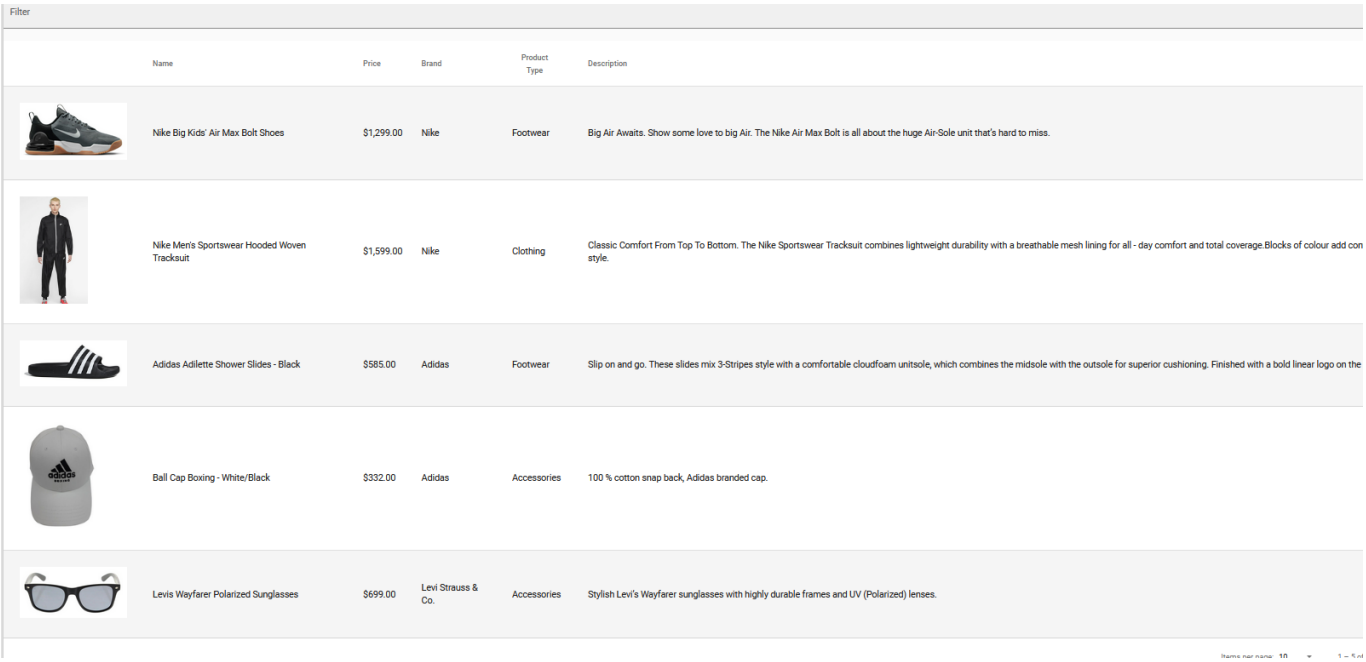


Fig. 5

- <u>Product Listing page</u>:
  - The Product Listing page should display the Products from the database retrieved via the API in a tabular format (Fig. 6)
  - The Product columns to display in the table are the **Image, Name**, **Price, Description**, the **Brand name** linked to the Product, and the **Product type** linked to the Product.
  - The User should be able to sort the products by **Name**, **Price**, **Brand**, **Product type**, and **Description** columns. I.e., by any of the columns mentioned in ascending or descending order.
  - The User should also be able to filter the product list by checking whether the filter text exists in either of the following columns: **Name**, **Price**, **Brand**, **Product type**, and **Description** (Fig. 7)**.**
  - Pagination should be incorporated to allow the User to display products as **3**, **5**, or **10** *Items per page* (Fig. 8)
  - The **Price** should be displayed as a monetary value (**Dollar**) allowing for two decimal points.



Filter

	Name	Price	Brand	Product Type	Description
	Nike Big Kids' Air Max Bolt Shoes	$1,299.00	Nike	Footwear	Big Air Awaits. Show some love to big Air. The Nike Air Max Bolt is all about the huge Air-Sole unit that's hard to miss.
	Nike Men's Sportswear Hooded Woven Tracksuit	$1,599.00	Nike	Clothing	Classic Comfort From Top To Bottom. The Nike Sportswear Tracksuit combines lightweight durability with a breathable mesh lining for all - day comfort and total coverage.Blocks of colour add con style.
	Adidas Adilette Shower Slides - Black	$585.00	Adidas	Footwear	Slip on and go. These slides mix 3-Stripes style with a comfortable cloudfoam unitsole, which combines the midsole with the outsole for superior cushioning. Finished with a bold linear logo on the
	Ball Cap Boxing - White/Black	$332.00	Adidas	Accessories	100 % cotton snap back, Adidas branded cap.
	Levis Wayfarer Polarized Sunglasses	$699.00	Levi Strauss & Co.	Accessories	Stylish Levi's Wayfarer sunglasses with highly durable frames and UV (Polarized) lenses.

Fig. 6

Filter
sungl

	Name	Price	Brand	Product Type	Description
	Levis Wayfarer Polarized Sunglasses	$699.00	Levi Strauss & Co.	Accessories	Stylish Levi's Wayfarer sunglasses w (Polarized) lenses.

Items per page: 10

Fig. 7

	Name	Price	Brand	Product Type	Description
	Adidas Adilette Shower Slides - Black	$585.00	Adidas	Footwear	Slip on and go. These slides mix 3-Stripes style with a comfortable cloudfoam unitsole, which combines the midsole witl cushioning. Finished with a bold linear logo on the side.
	Ball Cap Boxing - White/Black	$332.00	Adidas	Accessories	100 % cotton snap back, Adidas branded cap.
	Levis Wayfarer Polarized Sunglasses	$699.00	Levi Strauss & Co.	Accessories	Stylish Levi's Wayfarer sunglasses with highly durable frames and UV (Polarized) lenses.

Items per page: 3 1 – 3 of 5

Fig. 8

- <u>Add Product page</u>:
  - The Add Product page should allow the User to add new Product details and upload them to the database via the API.
  - The Add Product page requires **validation on all the controls** to add a Product (Fig. 9).
  - The Product columns and controls to display on the **Form** are the **Upload File** button**, Name**, **Price, Description**, **Brand name**, **Product type name**, and Submit button (Fig. 10).
  - The Price should only allow for numerical (including decimal) values.
  - The **Brand** and the **Product Type** Select controls should display the Brand Names and Product Type Names from the respective tables in the database**.** When the **Product** is submitted and saved to the database, the Brand Id and Product Type Id must be stored in the **Product** table.
  - When the User has successfully created the new Product, they must be redirected to the Product Listing page with the following notification message "**<<your product name captured>> created successfully**". (Fig. 11).

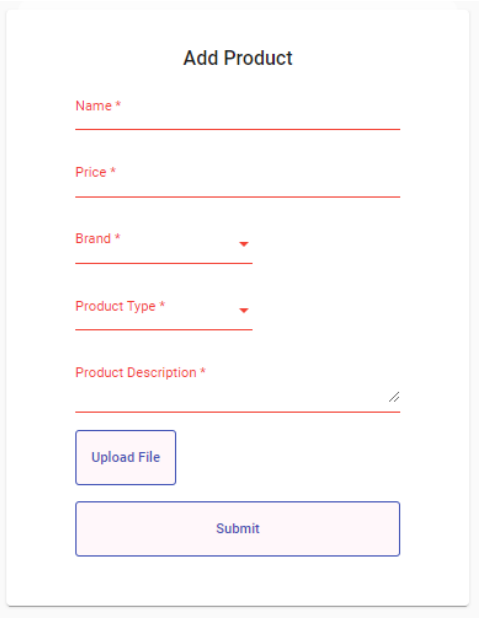

Add Product
Name *
Price *
Brand *
Product Type *
Product Description *
Upload File
Submit

Fig. 9

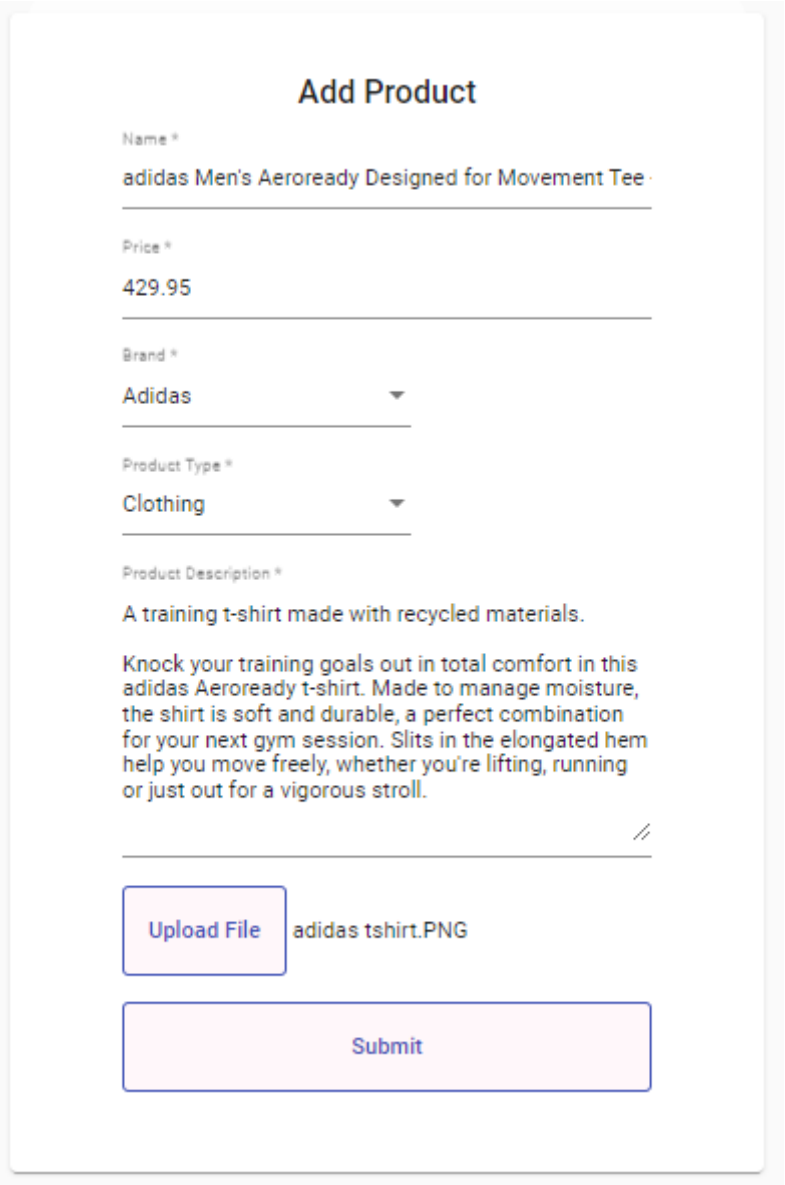

Add Product
Name *
adidas Men's Aeroready Designed for Movement Tee
Price *
429.95
Brand *
Adidas
Product Type *
Clothing
Product Description *
A training t-shirt made with recycled materials.
Knock your training goals out in total comfort in this adidas Aeroready t-shirt. Made to manage moisture, the shirt is soft and durable, a perfect combination for your next gym session. Slits in the elongated hem help you move freely, whether you're lifting, running or just out for a vigorous stroll.
Upload File
adidas tshirt.PNG
Submit

Fig. 10

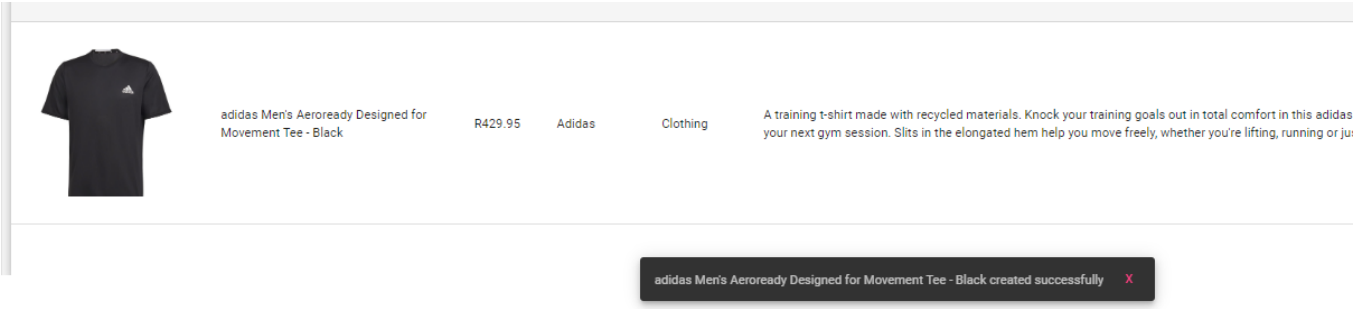


Fig. 11

- <u>Reporting page</u>:
  - The product reporting must display 2 bar graphs. One for the *Product count grouped by Brands*, and the other for the *Product count grouped by Product Type* (**Fig. 12**). <u>Note: the charts design must be identical. In other words, the same labels, y-axis numbers, bar colors (use "*#90E0EF*" and "*#00B4D8*"), etc</u>.
  - On the same page, you must display the "**Active Products Report**" based on the "*isActive*" state. The content to display is a hierarchy of ***Products*** displayed under the relevant ***Product Type*** and **Brand** Combination (**Fig. 13 and 14**). <u>Note: You need to display this information with an Angular-Material Accordion, as displayed in **Figure's 13** and **14**.</u>

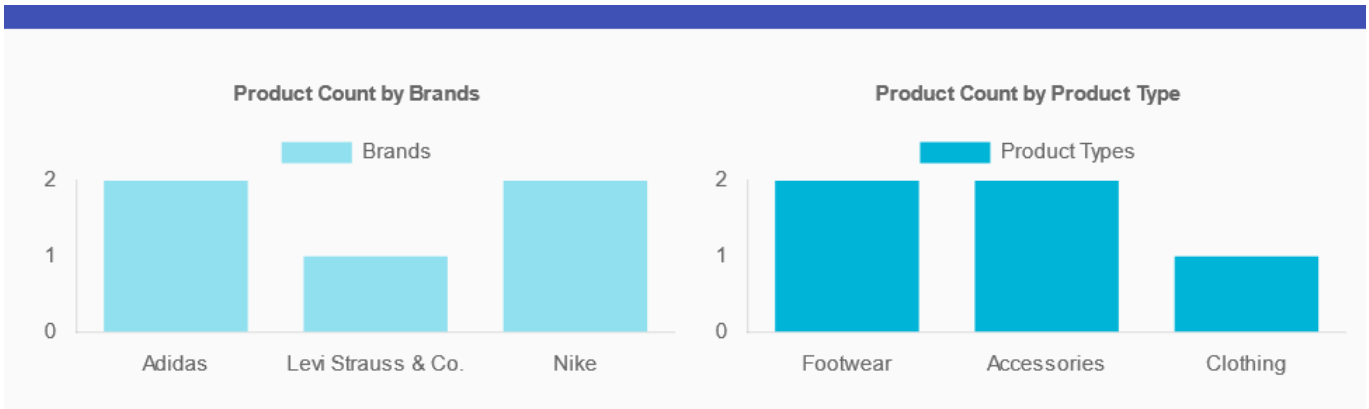


Fig. 12

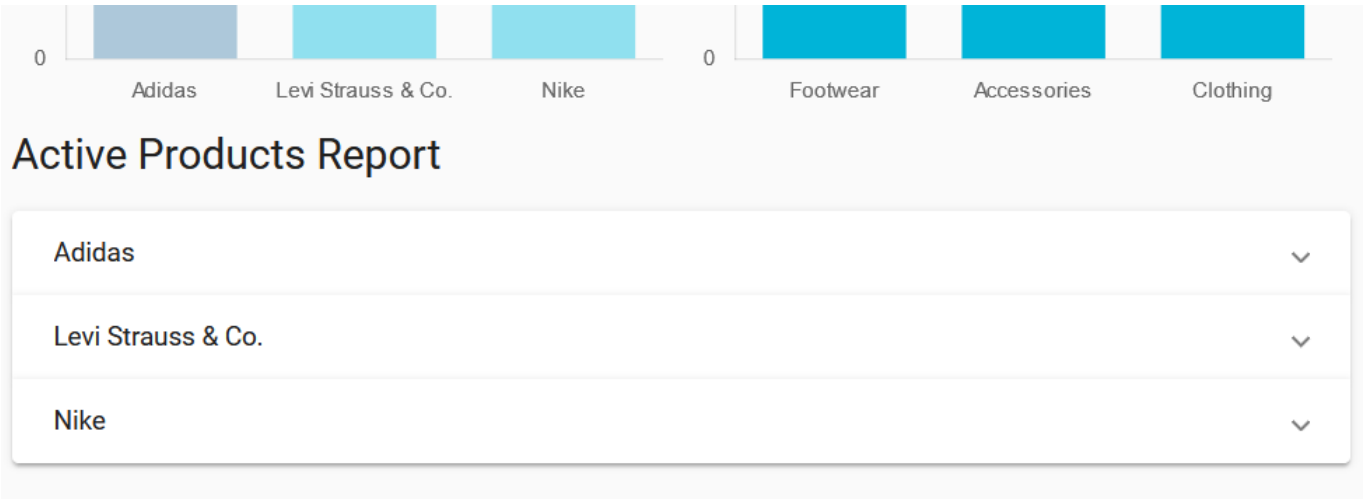


Fig. 13

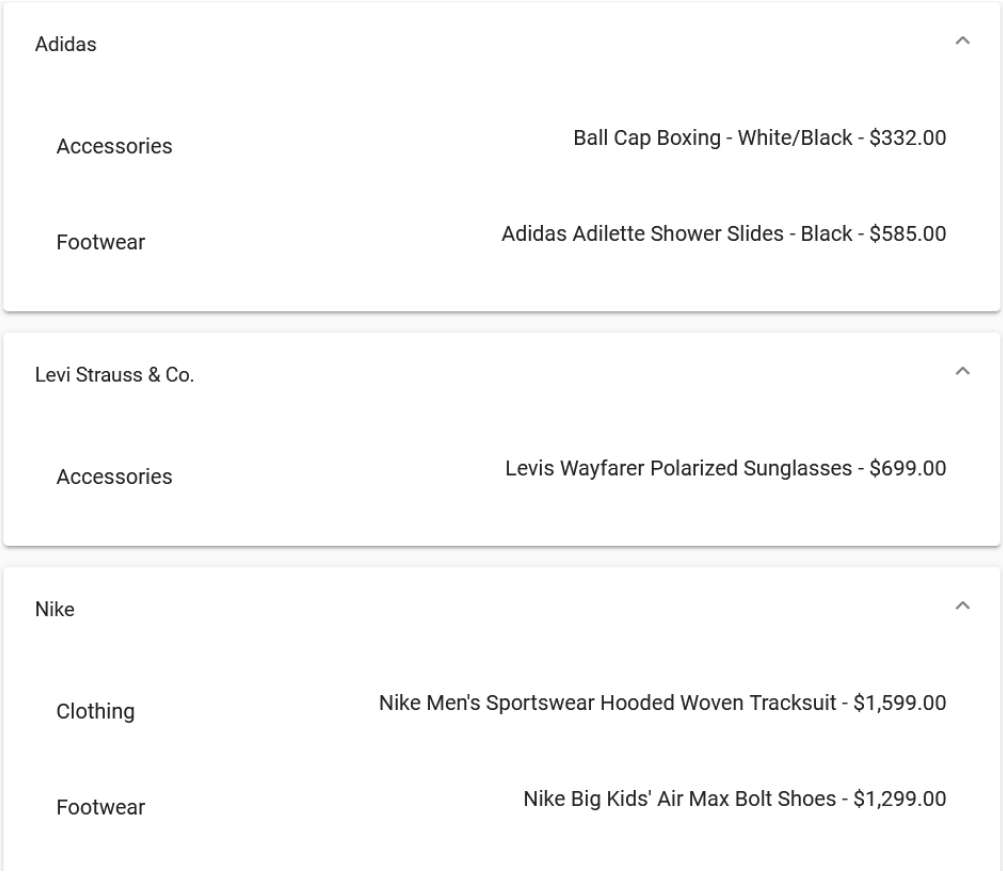


Fig. 14

ANGULAR AND API INSTALLATION AND CONFIGURATION:

- API:
  - An "**API Template**" has been created with the default configuration. In other words, Database Connection, the **Brand, Product, and Product type** entities, additional Middleware setup, and .Net Core installations to get you started.
  - Open the **Assignment03 .Net Core application** in Visual Studio 2022.
  - Once the application loads, open the "**appsettings.json**" file in Solution Explorer.
  - Change and save the **Server** location, pointing to your *SQL Server Server Name*). Alternatively, you can replace the server name with a period (.). See the example below.
  - Example:
    `optionsBuilder.UseSqlServer("Server=.;Database=INF354Assignment3;Trusted_Connection=True;MultipleActiveResultSets=True");`
  - Next, open the Package Manager Console (**View** > **Other Windows** > **Package Manager Console**) and run the following 2 commands individually to create the database tables from the abovementioned entities.
    - add-migration initial
    - update-database
  - The **INF354Assignment3 MS SQL Server 2022 database will create the relevant tables**.
  - Next, run the "**SqlDataCodeScript.sql**" script in **MS SQL Server 2022** to populate the **Products**, **Brands**, and **Product Types** with the initial data.
  - Now, run the API and have it running when you are trying to connect your Angular App to it. In other words, the API and the Angular App must be running for the application to work correctly.
- Angular:
  - You must create the Angular app yourself. I.e., there is no template to be shared. However, you can use any previous lectures and assignment source code shared with you, including internet services and tools. For example, some aspects of the Angular II lecture source code will be helpful in this Assignment. The same applies to all other source codes shared previously across lectures.
  - To run the application, type "**ng serve**" in the terminal window.

SUGGESTIONS:

- For the API, you will likely have 3 controllers (*the Authentication, Store and Report controllers with endpoints (functions) to talk to the database and Angular App*).

    - For example, an **AuthenticationController** with at least 2 endpoints (functions) to **Login (POST)**, and **Register (POST)**.
    - For example, a **StoreController** with at least 4 endpoints (functions) to **AddProduct (POST)**, **ProductListing (GET), GetBrands (GET), and GetProductTypes (GET)**.
    - For example, a **ReportController** with 1 endpoint (function) to Generate the *Products* report from the **Brands**, **ProductTypes**, and **Products** tables.
- You can **design your UI** *any way you want*, **so long it has all the controls and output required as specified in the Standard Requirements**.
- You can **develop your API** *any way you want*, **so long as it can perform the functionality required as specified in the Standard Requirements**.

RUBRIC:

Your assignment submission will be marked according to the following rubric:

Program (50 pts)	**(Exceptional)**	**(Very good)**	**(Good)**	**(Satisfactory)**	**(Poor)**	**(Very poor)**
Program Execution	The program executes correctly with no syntax or runtime errors. *I.e. the program has no execution issues.* (10)	The program executes with one or two syntax or runtime errors. *E.g. the program loads with no crashing but displays minor bugs in the debugger.* (8)	The program executes with a few syntax or runtime errors. *E.g. A couple of runtime errors and/or the program crashes at one screen/section.* (6)	The program executes with many syntax or runtime errors. *E.g. A few runtime errors and/or the program crashes at two screens/sections.* (5)	The program executes with major errors. *E.g. The program can execute, however, it is plagued with runtime or syntax errors, or the program keeps crashing during use.* (3)	The program does not execute. *I.e. The application fails to run.* (0)
Program Functionality	Program functionality is in line with the requirements. *I.e. the program has all the correct functionality implemented.* (10)	Program functionality has one minor inconsistency. *E.g. One of the functional requirements is incorrect.* (8)	Program functionality has a few minor inconsistencies. *E.g. Two of the functional requirements are incorrect or one is missing.* (6)	Program functionality has many inconsistencies. *E.g. Some of the functional requirements are incorrect or half is missing.* (5)	Program functionality has major inconsistencies. *E.g. Most of the functional requirements is incorrect or missing.* (3)	Program functionality is missing. *E.g. None of the functionality works or all the functionality is missing.* (0)
Program Output	The program displays correct output in line with the requirements. *I.e. It produces the same output as required.* (10)	The program has one or two very minor output discrepancies. *I.e. It produces output with barely noticeable inconsistencies. E.g. one or two formatting issues.* (8)	The program has a few output discrepancies. *I.e. It produces output with easily noticeable inconsistencies. E.g. The program does not return some of the data or there are a few formatting issues.* (6)	The program has many output discrepancies. *I.e. It produces output with many noticeable inconsistencies. E.g. The program does not return half of the data or there are plenty of formatting issues.* (5)	The program has major output discrepancies. *I.e. The output is plagued with inconsistencies. E.g. The program does not return most of the data or there are substantial formatting issues.* (3)	Output is incorrect. *E.g. The program does not provide any of the requested output or all the formatting is not as requested in the requirements.* (0)
Program Interface (UI)	The program interface is professionally done. I.e. *The interface is implemented correctly and looks very good.* (5)	The program interface is done well. *I.e. The interface is implemented correctly and looks good. E.g. One or two styling/layout issues.* (4)	N/A	The program interface is good enough. *I.e. The interface is implemented correctly and looks okay. E.g. A few styling/layout issues.* (3)	The program interface is poorly done. *I.e. The interface is mostly incorrect or looks poorly done. E.g. The layout is mostly incorrect or has plenty of styling issues.* (2)	The program interface is very poor. *I.e. The interface is entirely incorrect or is very poorly done. E.g. The layout is completely incorrect or the styling is missing.* (0)
Code Readability	The program code is well organized and makes good use of white space. Variables have	Program code is organized and makes use of white space. Variables have descriptive names.	N/A	Program code is mostly organized and makes use of white space. Most variables have descriptive	Program code is somewhat organized, and not easy to read and understand. *E.g. There are plenty of variable naming convention issues*	Program code is difficult to read. *E.g. Variable naming conventions are*

Program (50 pts)	(Exceptional)	(Very good)	(Good)	(Satisfactory)	(Poor)	(Very poor)
	descriptive names. I.e. *There is nothing to fault on.* (5)	*E.g. There are one or two variable naming convention issues or white space issues.* (4)		names. *E.g. There are a few variable naming convention issues or program code organization that could be improved.* (3)	*or the code is challenging to follow.* (2)	*missing or the code is hard to follow.* (0)
Video Demonstration	The program is exceptionally well presented. I.e. *The student demonstrated and displayed all the required functionality, output, interfaces, and code.* (10)	The program is well presented. *E.g. The student demonstrated and displayed all the required functionality, output, interfaces, and code. However, one of the descriptions or illustrations was lacking.* (8)	The program presentation is good. *E.g. The student demonstrated and displayed most of the required functionality, output, interfaces, and code. However, two of the functionality, output, interfaces and code descriptions or illustrations were lacking or missing.* (6)	The program presentation is adequate. *E.g. The student demonstrated and displayed most of the required functionality, output, interfaces, and code. However, a few to half of the functionality, output, interfaces and code descriptions or illustrations were lacking/missing.* (5)	The program is presented poorly. *E.g. The student demonstrated and displayed a few of the required functionality, output, interfaces, and code. However, most functionality, output, interfaces, and code descriptions or illustrations were lacking/missing.* (3)	The program has barely been presented or has been presented very poorly. *E.g. The student failed to demonstrate and display the required functionality, output, interfaces, and code or it was missing.* (0)