Assignment 3
Out of 50 Marks

DUE: 19 May 2025

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., 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: 19 May 2025

- 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 has commissioned the development of a proof-of-concept application utilizing Angular for the front end and .NET 8 Web API for the back end. The core functionalities required include user registration, login, and the ability to record and browse inventory products.
- You are tasked with building the back-end services using .NET 8 API and developing the front-end interface using Angular.
- The application must support the creation and retrieval of product records, which are to be stored in a SQL Server 2019 database. Additionally, the system must incorporate authentication features, enabling users to register and securely log in.
- Upon launching the application, the Login page should serve as the landing page. Navigation to other sections of the application must be managed through Angular routing, with access governed according to the restrictions and guidelines outlined under the "Standard Requirements" section.

STANDARD REQUIREMENTS:

- Login page:
  - The Login page must require the User to enter a Username (which must be a valid email address) and a Password. Login functionality should be disabled if either the Username or Password is not provided (refer to Fig. 1).
  - If the User clicks the link labeled "Don't have an account? Register here", they must be redirected to the Register page (refer to the Register page section for details).
  - Upon entering valid credentials, the User must be redirected to the Product Listing page (see the Product Listing page section for further information).
  - After a successful login, a side navigation menu containing Product Listing, Add Product, and Logout options must be displayed, allowing users to access the associated functionalities (refer to Fig. 2).
  - Selecting the Logout option from the side menu must log the User out of the system, hide the side navigation menu, and redirect the User back to the Login page (refer to Fig. 3).

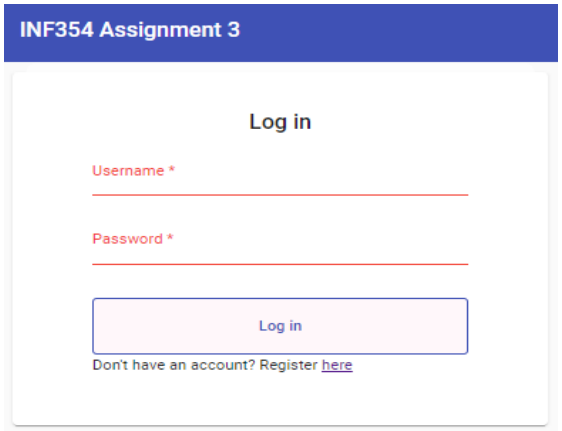

Fig. 1

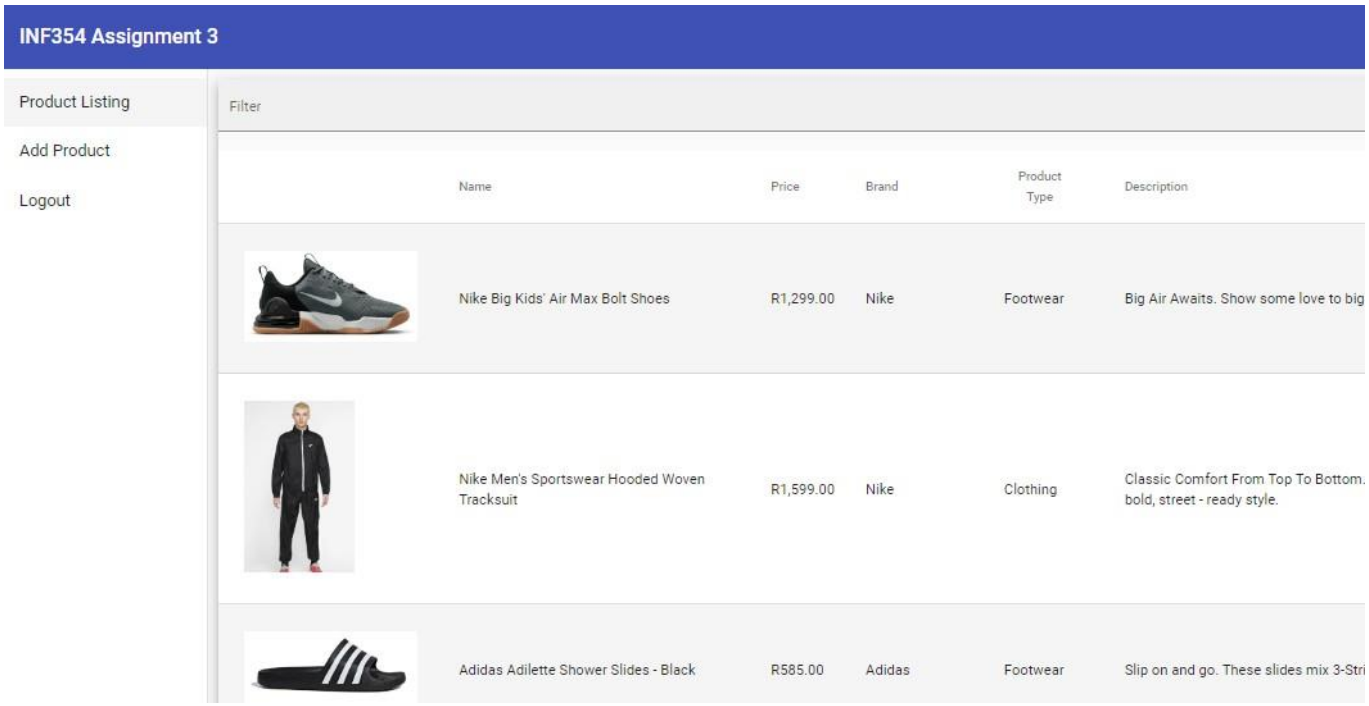

Fig. 2

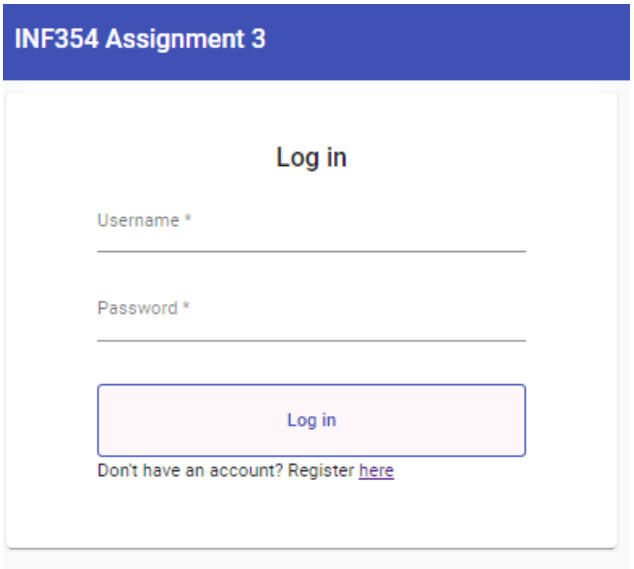

Fig. 3

- 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.

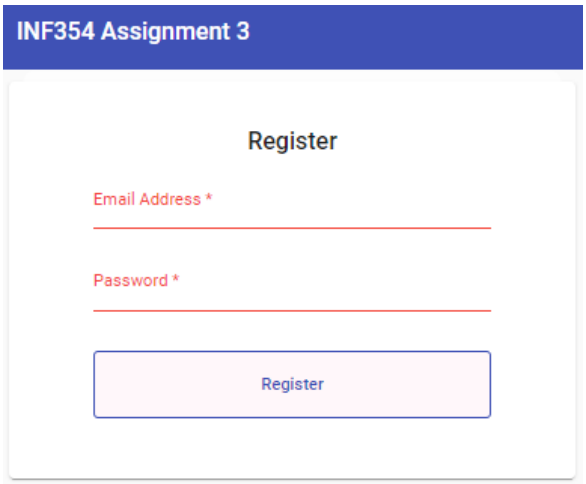

Fig. 4

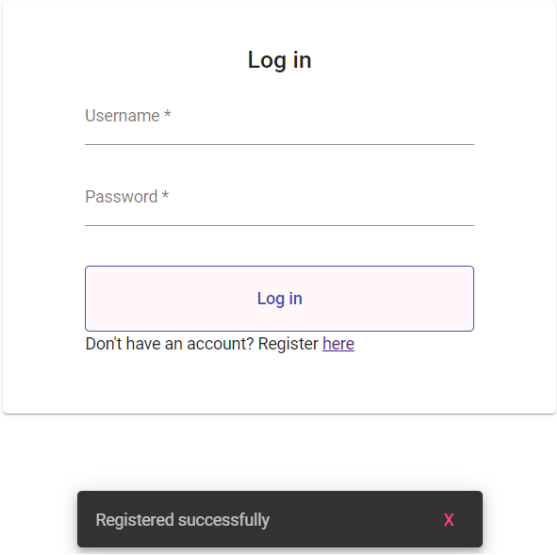

Fig. 5

- <u>Product Listing page</u>:
  - The Product Listing page must retrieve and display the list of products from the database through the API, presenting them in a tabular format as illustrated in Figure 6.
  - The table should display the following columns: Product Image, Product Name, Price, Description, Brand Name associated with the product, and the Product Type associated with the product.
  - The User must be able to sort the product listing based on any of the displayed columns — namely, Name, Price, Brand, Product Type, and Description — in either ascending or descending order.
  - Additionally, the User should be able to filter the product list by entering text, and the system must check whether the filter text exists in any of the following columns: Name, Price, Brand, Product Type, or Description as demonstrated in Figure 7.
  - Pagination functionality must be incorporated, enabling the User to select and display 3, 5, or 10 products per page, as shown in Figure 8.
  - The Price must be formatted and displayed as a monetary value in South African Rands (ZAR), with exactly two decimal points.

Filter

	Name	Price	Brand	Product Type	Description
	Nike Big Kids' Air Max Bolt Shoes	R1,299.00	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	R1,599.00	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 contrast for bold, street - ready style.
	Adidas Adilette Shower Slides - Black	R585.00	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 side.
	Ball Cap Boxing - White/Black	R332.00	Adidas	Accessories	100 % cotton snap back, Adidas branded cap.

Fig. 6

Filter
Ball Cap

	Name ↓	Price	Brand	Product Type	Description
	Ball Cap Boxing - White/Black	R332.00	Adidas	Accessories	100 % cotton snap back, Adidas branded cap.

Items per page: 10 1 – 1 of 1

Fig. 7

	Name	Price	Brand	Product Type	Description
	Nike Big Kids' Air Max Bolt Shoes	R1,299.00	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	R1,599.00	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 contrast for bold, street - ready style.
	Adidas Adilette Shower Slides - Black	R585.00	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 side.

Items per page: 3 1 – 3 of 5

Fig. 8

- Add Product page:
  - The Add Product page should enable the user to input new product details and upload them to the database through the designated API.
  - Validation must be implemented on all input controls associated with adding a product (refer to Fig. 9).
  - The form on the Add Product page must include the following controls and display fields: Upload File button, Name, Price, Description, Brand Name, Product Type Name, and a Submit button (refer to Fig. 10).
  - The Price field must strictly accept numerical values, including decimal numbers, to ensure accurate pricing information.
  - The Brand and Product Type select controls should dynamically populate their options with the corresponding Brand Names and Product Type Names retrieved from their respective database tables. Upon submission, the selected Brand Id and Product Type Id must be stored in the Product table alongside the new product record.
  - After the user successfully creates a new product, they must be redirected to the Product Listing page and presented with a confirmation notification stating: "<<your product name captured>> created successfully." (Refer to Fig. 11).

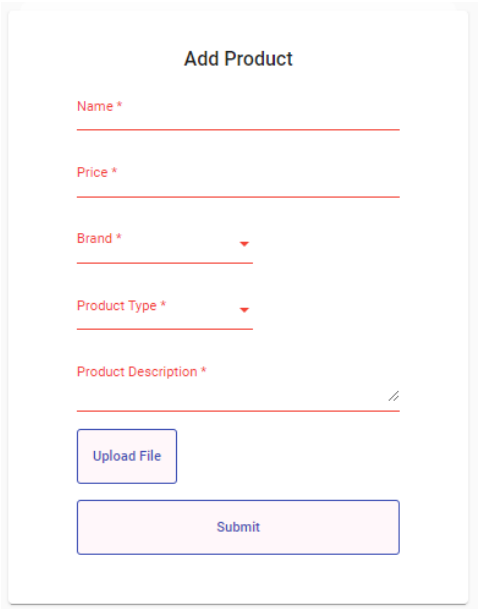

Fig. 9

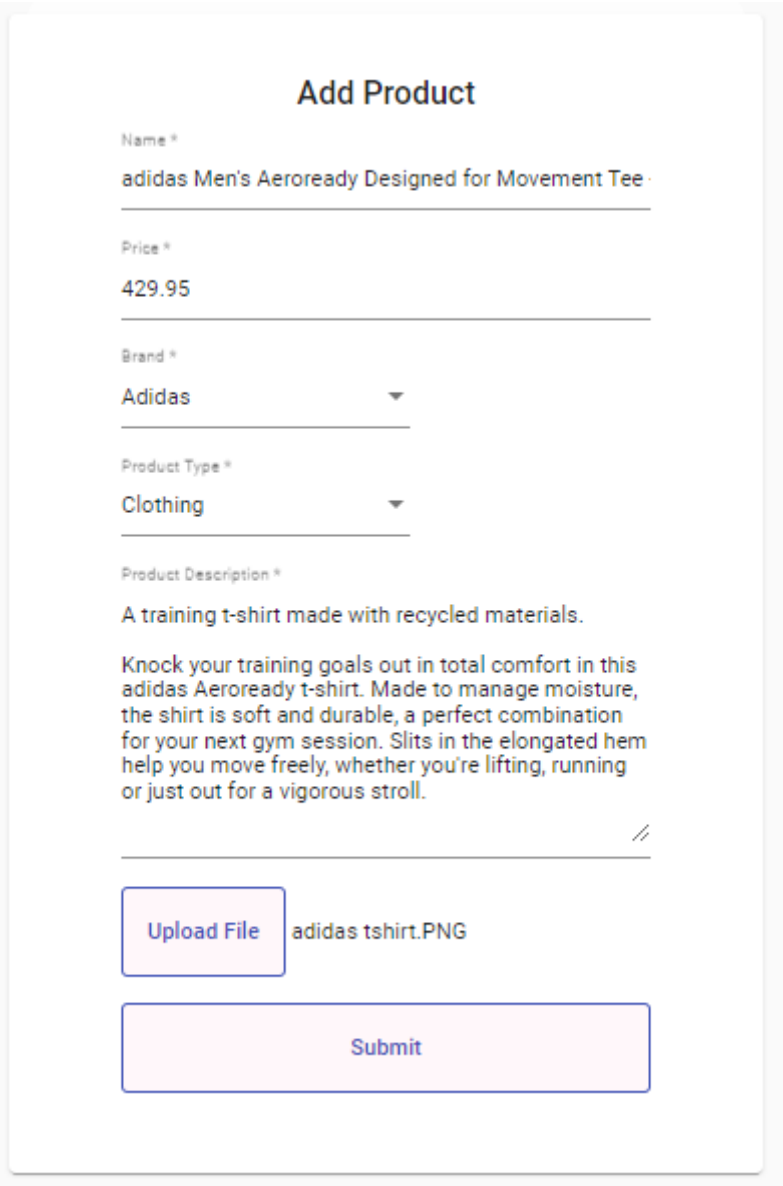

Fig. 10

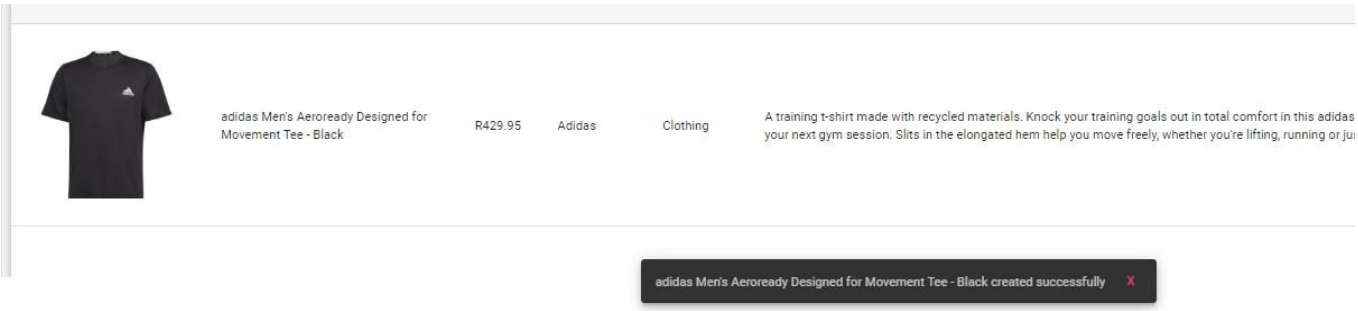

Fig. 11

ANGULAR, REPORTING AND API 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's name with a period (.). See the example below.
- Example:
  `optionsBuilder.UseSqlServer("Server=UP957721\\MSSQLSERVER01;Database=INF3542025Assignment3;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 **INF3542025Assignment3 MS SQL Server 2019 database will create the relevant tables**.
- Next, run the "**SqlDataCodeScript.sql**" script in **MS SQL Server 2019** 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.
- ProductDashboard page:
  - The product dashboard must display 2 pie charts. One for the Product count grouped by Brands, and the other for the Product count grouped by Product Type (Fig. 12).
  - On the same page, you must display the "Top 10 most expensive products" based on the product price. The columns to display are the product name (Name), the product price (Price), product brand (Brand), product type (Type), and the product description (Description) (Fig. 13).
  - Note: Your top 10 products will not be the same as that displayed.

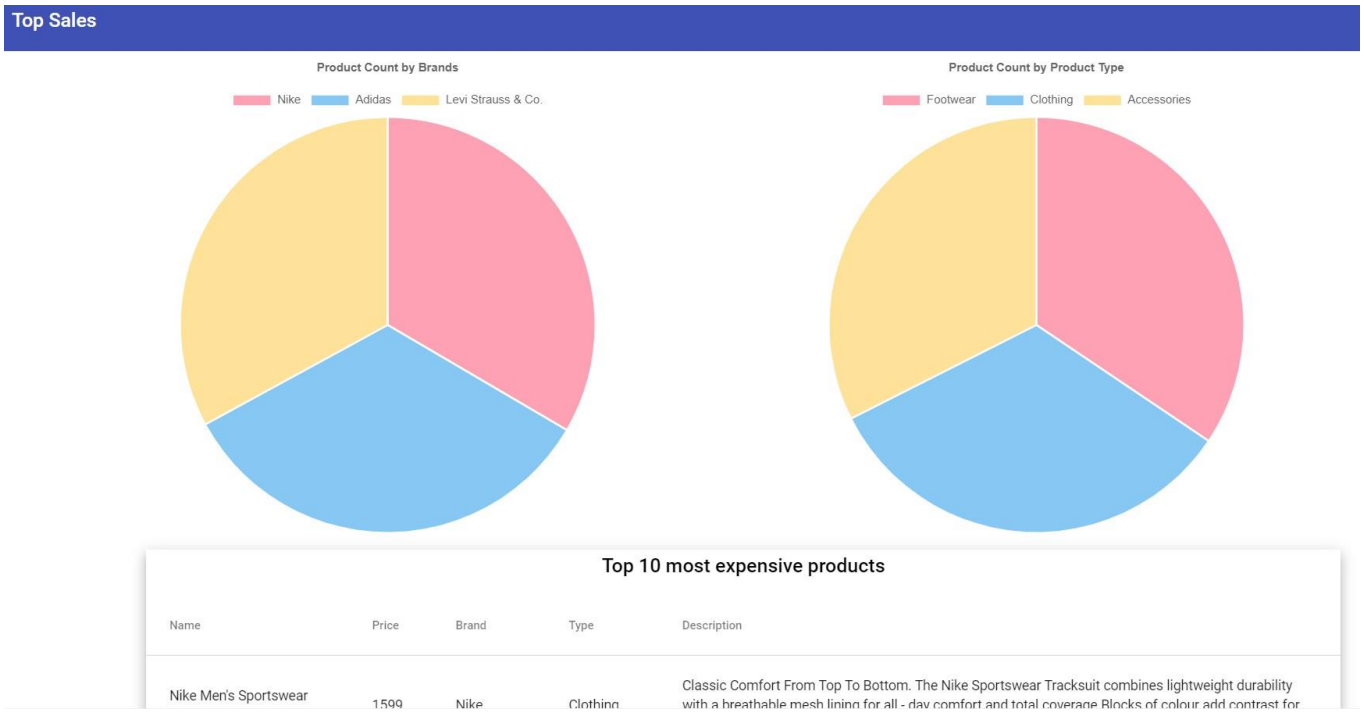

Fig. 12

Top 10 most expensive products

Name	Price	Brand	Type	Description
Nike Men's Sportswear Hooded Woven Tracksuit	1599	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 contrast for bold, street - ready style.
Product 414	1500	Adidas	Accessories	Description for Product 414
Product 308	1499	Adidas	Footwear	Description for Product 308
Product 920	1499	Adidas	Clothing	Description for Product 920
Product 49	1498	Levi Strauss & Co.	Accessories	Description for Product 49
Product 291	1498	Levi Strauss & Co.	Accessories	Description for Product 291
Product 353	1497	Nike	Clothing	Description for Product 353
Product 430	1493	Nike	Clothing	Description for Product 430
Product 175	1489	Levi Strauss & Co.	Accessories	Description for Product 175
Product 325	1489	Nike	Clothing	Description for Product 325

Fig. 13

SUGGESTIONS:

- For the API, you will likely have 2 controllers (*the Authentication and Store 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)**.
- 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)