**Features of the application:** 1. User Registration

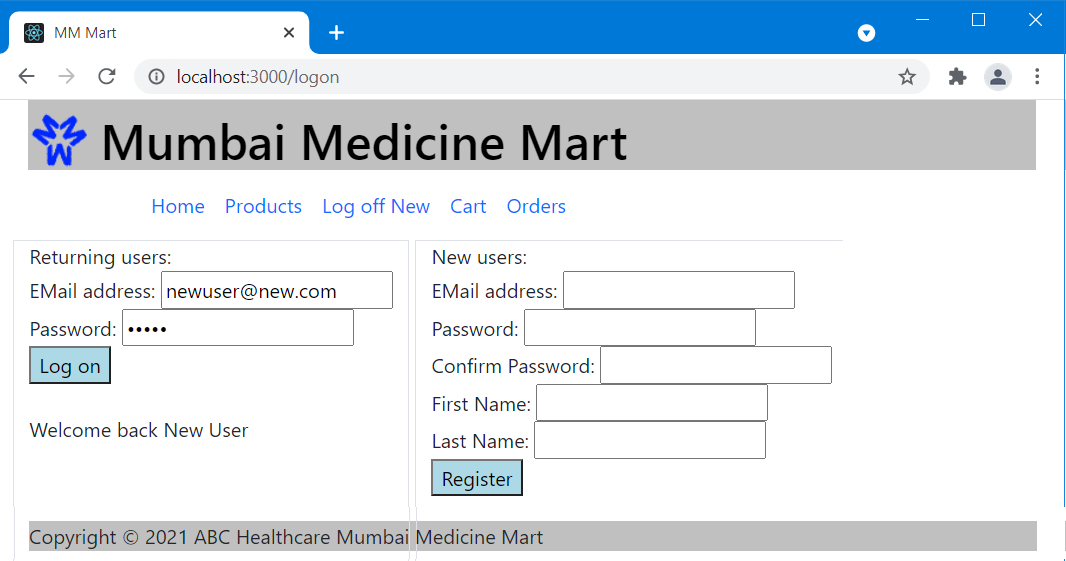
Registration is accomplished by choosing “Log on” from the menu bar and filling out the right-hand form:

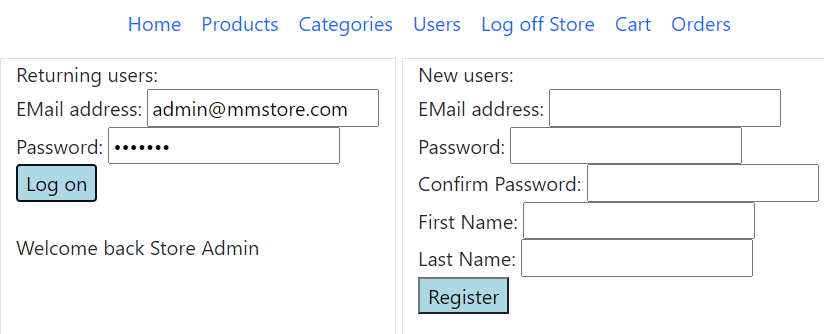


If the email address is valid and not already in use, a password is specified, the passwords match, and a name was given then a new user is created in the database and “Welcome <first> <last>” is displayed.

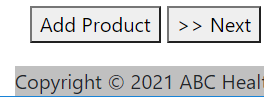
**Features of the application:** 2. User Login

**Features of the application:** 3. Admin Login

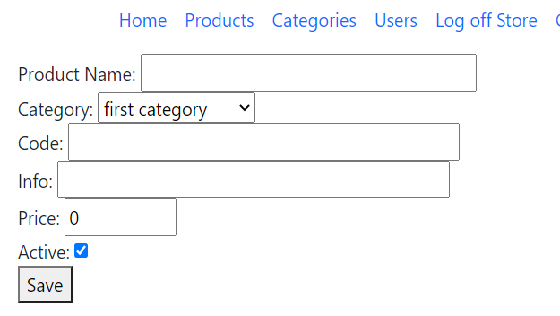
The single logon option handles both user and admin logons; the back-end verifies the access level set for the user in the database at logon and on all requests. For a standard user the front-end displays a limited menu after a successful logon:

When an admin user logs on the front-end displays an extended menu that includes the Categories and Users options which standard users are not permitted to access.

**Features of the application:** 4. Add/Update medicines to portal

When an admin user selects the Products menu option the list of current products is shown and a button appears at the bottom of the page to allow adding a new product:

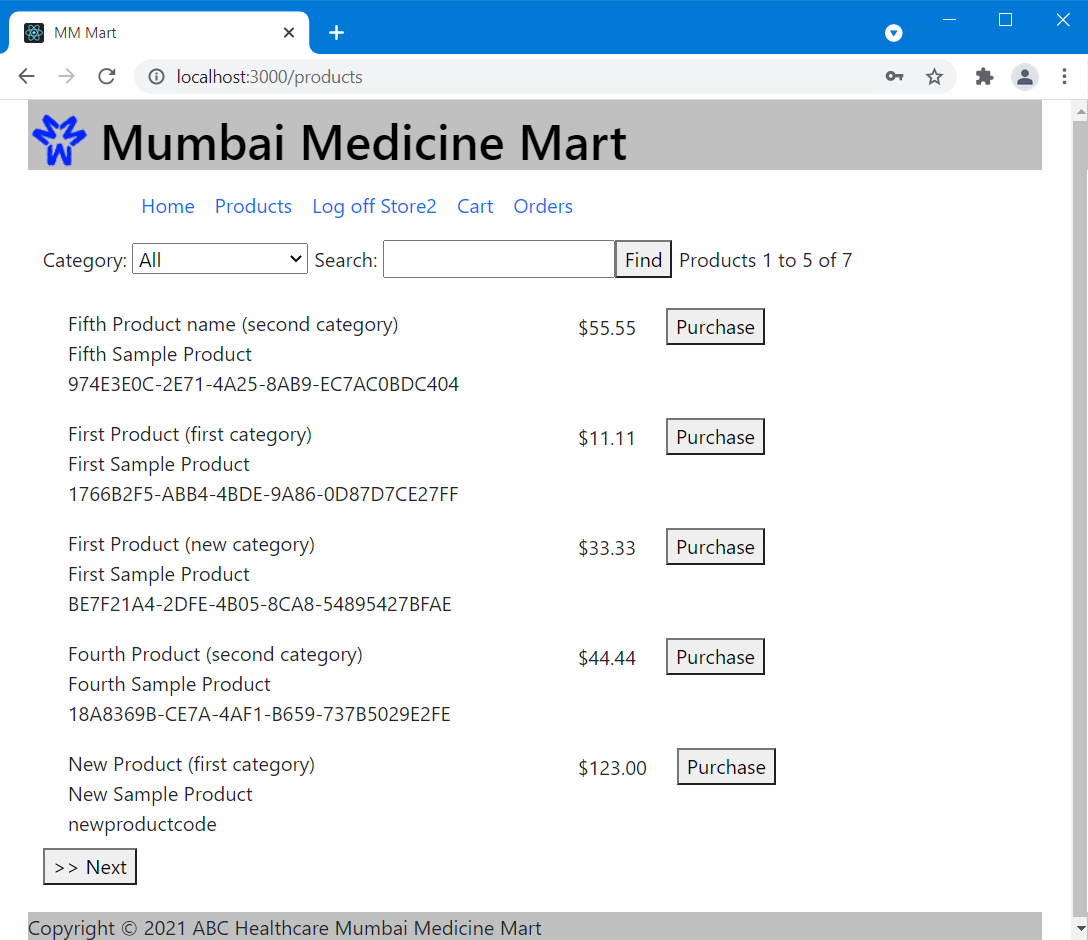
When that button is clicked the user is taken to a page that allows entry of a new item:



The category is selected from the drop-down list of available categories; other fields are entered as text. If the product name and code are unique then the item is added to the database. If the Active option is unchecked then this item will not be available for purchase.

**Features of the application:** 5. Browse through medicines.

**Features of the application:** 6. Search medicines.

The Products option in the menu bar shows the list of products sorted by name. By default it shows all products from all categories but the category drop-down can be used to filter the list to any desired category. The user can also enter any part of the product name or product code to limit the list to items matching the entered text:

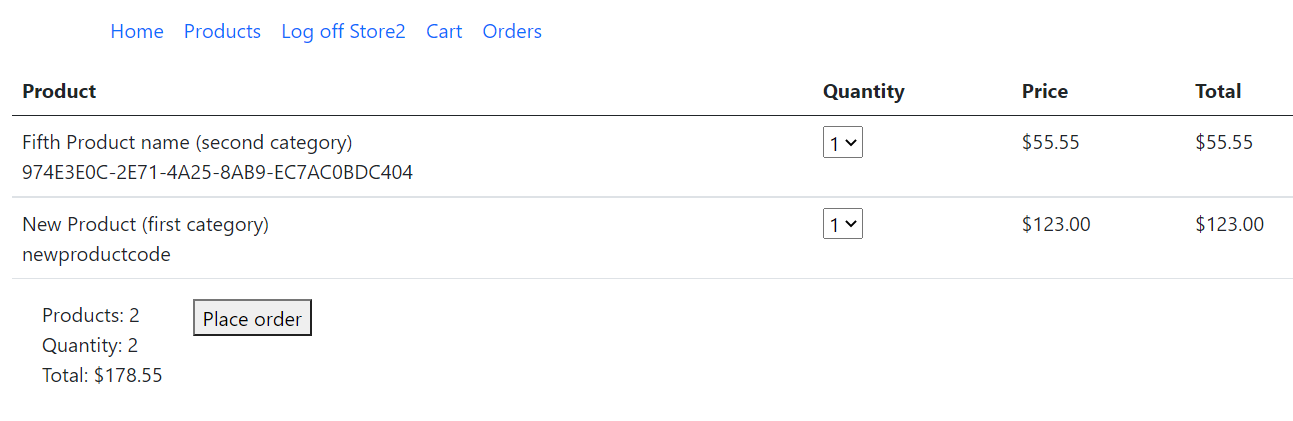
The application shows five products at a time and adds BACK and/or NEXT buttons at the bottom to allow the user to move through the pages. Each page change is a call to the back-end API to retrieve the appropriate items to display. If the user is logged on the PURCHASE buttons appear and allow the user to add the item to the cart. If the user has admin rights then an EDIT option will also appear to allow editing of the products.

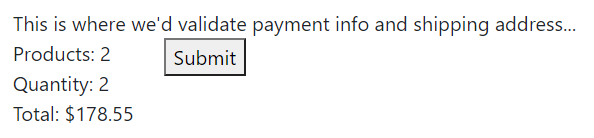
**Features of the application:** 7. Sort medicines based on name, category.

This requirement was unclear as it makes little sense to sort by category; the submitted product sorts by name and allows filtering by category.

**Features of the application:** 8. Manage Cart

**Features of the application:** 9. Place order

Selecting the Cart option from the menu displays the shopping cart for the current user. The menu item is only available if a user is currently logged on. The user can edit the quantities to adjust what they want to order. Setting a quantity to zero will cause it to not be included in any order.

Clicking the Place Order button will take the user to a summary screen to confirm the order placement. This app does not currently request payment or shipping data but that would be added here.

The number of products on these screens shows the total number of unique products in the cart or on the order. The Quantity field indicates how many total items are being ordered. If a user orders 2 of product #1 and 3 of product #2 there would be 2 products with a quantity of 5.

After submitting, the user will be given the order number as confirmation:



**Recommended Technologies:**

1. Database management: SQLServer

2. Back-end logic: ASP.NET Web API

3. Front-end development: React, HTML/CSS, Bootstrap.

4. Testing technologies: NUnit/XUnit

5. DevOps and production technologies: Git, GitHub, Docker

The project uses a C# ASP.Net Web API back-end to query and update tables in SQL server and has NUnit testing examples for the API calls. The front-end is React. All code is available at <https://github.com/bitbin42/FSDSection4>. Docker was not used as I do not have the resources available to implement that.

Comments from first submission:

Comment: The output screenshots in regards to Database management functionalities should be shared,

Response: **I have absolutely no idea what is meant by that**. The project submission included the SQL command to initialize the tables and sample data for testing. A screen shot of those same commands in SSMS would be pointless and, other than that, I don’t know what could be meant.

Comment: there was no output screenshot in regards to the use of any Testing technologies: NUnit/XUnit,

Response: added below

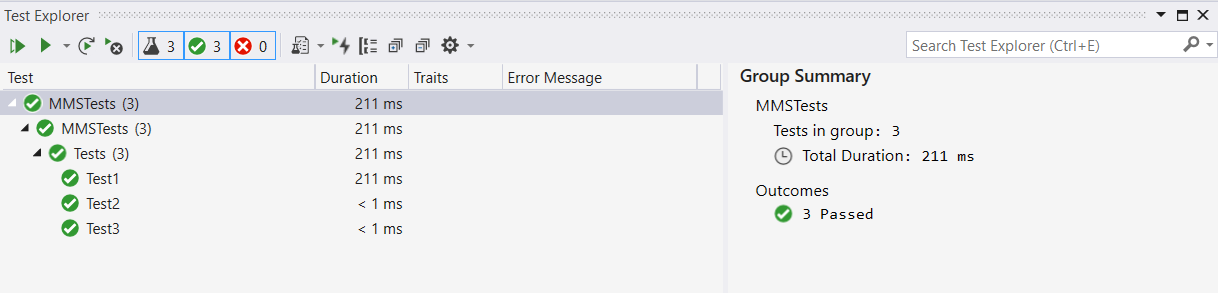
Comment: the output screenshots shared should be large and clear.

Response: screen shots were embedded n this Word document to make them easier to organize and to include related comments. **There were no specifications given regarding the size or format of screenshots in any of the requirements documents.**

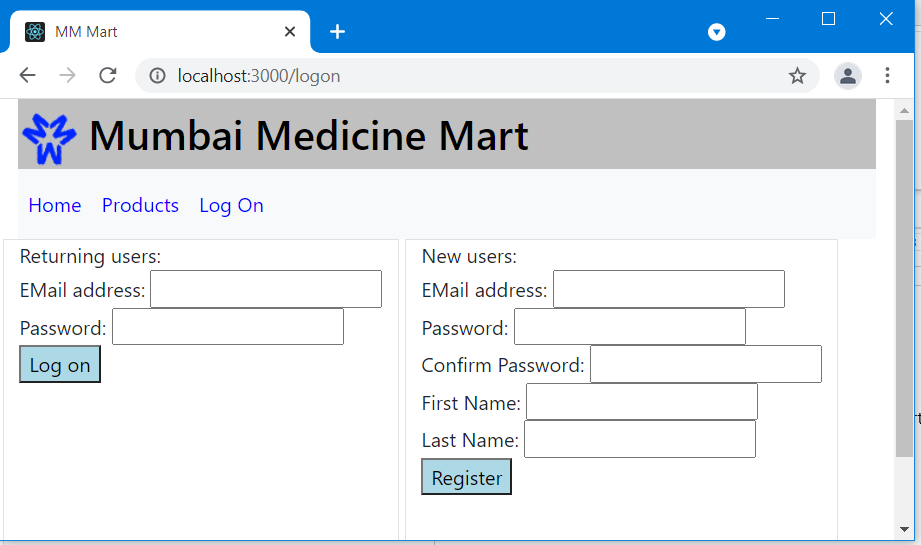
Comment: Any screenshot in regards to the project application reflecting the responsiveness and be user-friendly on mobile devices will be appreciated.

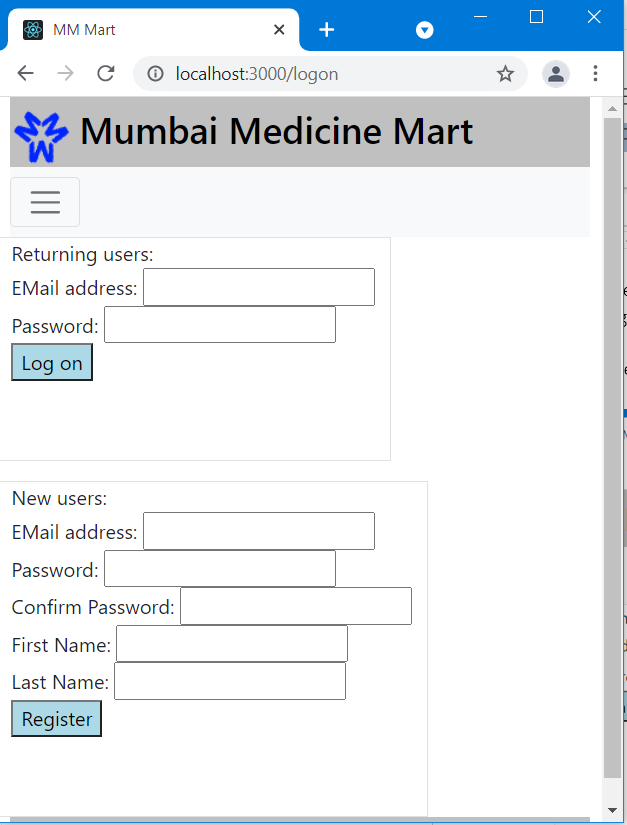
Response: The use of bootstrap was given a cursory mention in the class but not actually covered in any detail. It was not covered at all with regard to use from React. **Given that training was not provided it is not reasonable to include this as a requirement**. That said, the client does now reformat for small screens using react-bootstrap and a few screen shots are included below.

Nunit testing (after splitting the original test into 3 parts):

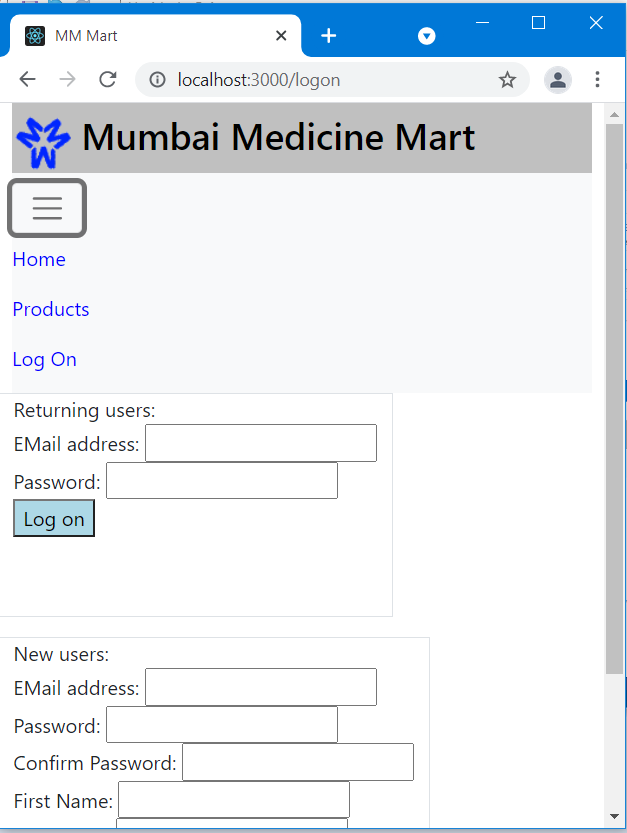


Screen views showing the menu converted to a “hamburger” and the remaining text wrapping to be readable on a small screen:

Large screen logon:

Small screen

logon:

Small screen after clicking the “hamburger”: