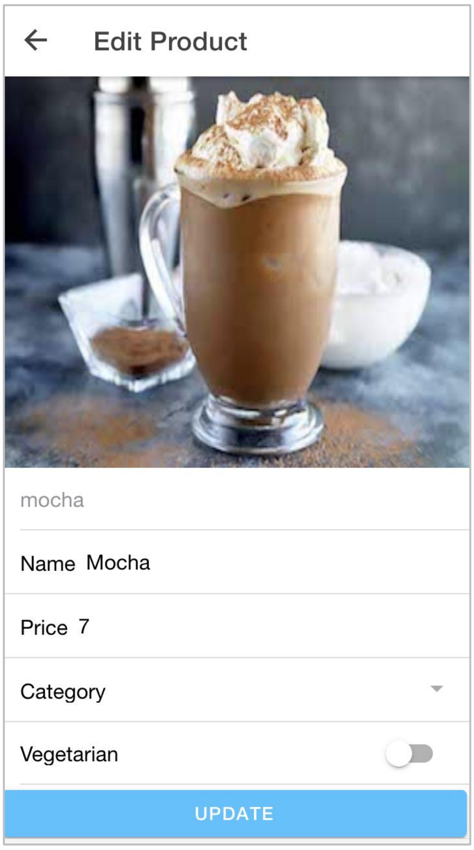
Practical 10: CRUD

Objectives:

Retrieve Firestore collections and documents. Modify data in Firestore by delete, update and add a new product.



Edit Product

mocha

Name Mocha

Price 7

Category

Vegetarian

Tasks:

1. Connect to Firebase
2. Read Collection (Products) from Firestore
3. Read Document (Product by id) from Firestore
4. Delete Product
5. Update Product
6. Add Product
7. Add Cart and Cart items

1 Connect to Firebase

1.1 Open folder in code editor

1. Open a web code editor such as **Visual Studio Code**.

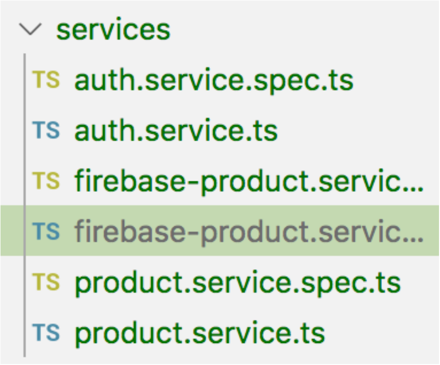
2. Select **File > Open Folder…** Select *skippyQ* folder.

3. Ensure that you have completed the previous practical to import the firebase config in *app.components.ts*.

1.2 Firebase Product Service

1. Using **Terminal**, generate a new service ***FirebaseProductService***.

```
ionic generate service shared/services/firebaseProduct
```



```
services
  auth.service.spec.ts
  auth.service.ts
  firebase-product.servic...
  firebase-product.servic...
  product.service.spec.ts
  product.service.ts
```

2. Open *src > app > shared > services > firebase-product.service.ts*. Import firebase.

firebase-product.service.ts

```
import firebase from 'firebase/app';
import 'firebase/firestore';
import 'firebase/storage';
```

2 Read Products from Database

2.1 Read Collection from Firestore

1. Open *src > app > shared > services > firebase-product.service.ts*.

   Here we are reading from the *'products'* collection in the database.

   Store the database path for *'products'* as a property so that we can easily change if required

firebase-product.service.ts

```
export class FirebaseProductService {
 private productsRef = firebase.firestore().collection("products");
```

2. Add `getProducts()` method.

firebase-product.service.ts

```
getProducts(): Observable<any> {
   return new Observable((observer) => {
     this.productsRef.onSnapshot((querySnapshot) => {
       let products: Product[] = [];
       querySnapshot.forEach((doc) => {
         let data = doc.data();
          let p = new Product(data['name'], data['price'], data['image'],
doc['id']);
         if (data['category']) p.category = data['category'];
         if (data['vegetarian']) p.vegetarian = data['vegetarian'];

         products.push(p);
       });
       observer.next(products);
     });
   });
 }
```

3. Use **Quick Fix** to add all the necessary imports.

firebase-product.service.ts

```
import { Product } from '../models/product';
import { Observable } from 'rxjs';
```

4. Open **Catalogue Page** *src > app > tab2 > tab2.page.ts*.

   Change ***ProductService*** to ***FirebaseProductService***.

   Use **Quick Fix** to import the class.

tab2.page.ts

```
constructor(private productService: FirebaseProductService) { … }
```

5. We can’t use the old way to get products anymore. Subscribe to the new service’s `getProducts()` method to get the data.

tab2.page.ts

```
constructor(private productService: FirebaseProductService) {

   ~~// this.products = this.productService.getProducts();~~

   this.productService.getProducts()
      .subscribe(data => {
        this.products = data;
     });

 }
```

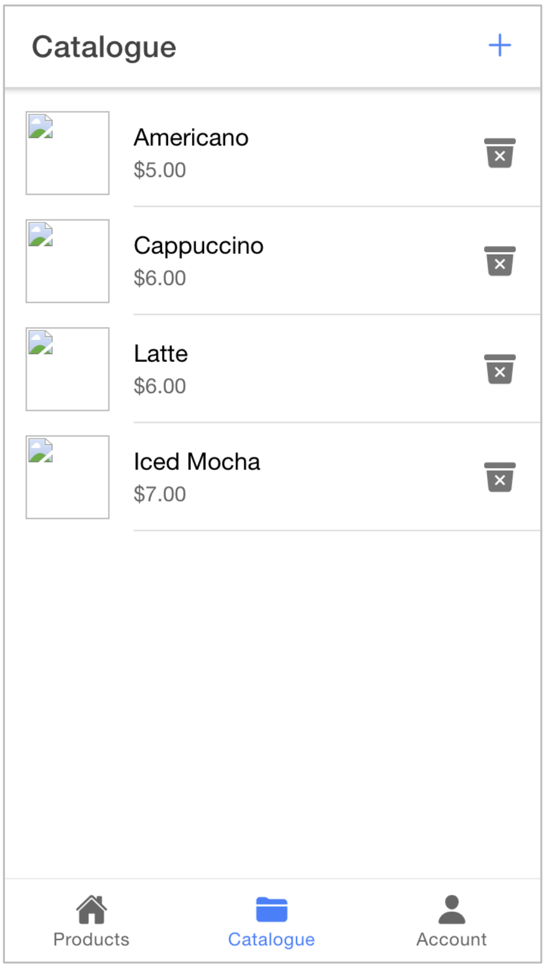
6. Comment off the delete() function. Will fix it later

```
delete(item: Product) {
    //this.productService.delete(item);
  }
```

7. Test the app in your computer browser using `ionic serve`.

   Go to **Catalogue Page**.

   You should see the products saved in the Database.



Catalogue

Americano
$5.00

Cappuccino
$6.00

Latte
$6.00

Iced Mocha
$7.00

2.2 Read Image from Firebase Storage

1. Open *src > app > shared > models > product.ts*.

   Add a new property `imagePath`.

product.ts

```
export class Product {
    name: string;
    price: number;
    image: string; // For displaying in <img>
    imagePath!: string; // Path for retrieving
    id: string;
    category: string;

    vegetarian: boolean;


    constructor( … ) { … }


}
```

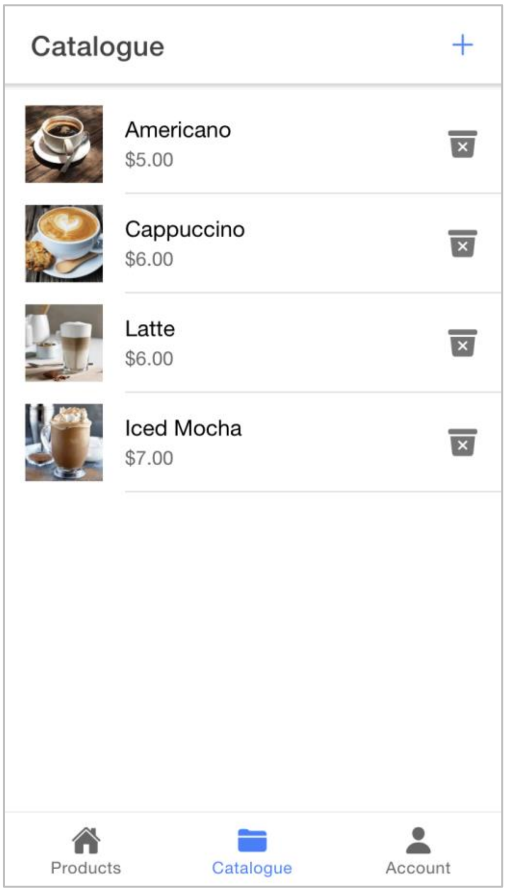
2. Open *src > app > shared > services > firebase-product.service.ts*.

Find the `getProducts()` method and add in the following codes to retrieve the image.

firebase-product.service.ts

```
 getProducts(): Observable<any> {

    return new Observable((observer) => {

      this.productsRef.onSnapshot((querySnapshot) => {

        let products = [];

        querySnapshot.forEach((doc) => {

          let data = doc.data();

          let p = new Product(data.name, data.price, data.image, doc.id);

          if (data.category) p.category = data.category;

          if (data.vegetarian) p.vegetarian = data.vegetarian;


          // If there's image, read from Firebase Storage
           if (data['image']) {
               p.imagePath = data['image'];
               const imageRef =
firebase.storage().ref().child(data['image']);
           imageRef.getDownloadURL()
              .then(url => {
                 p.image = url;
              }).catch(error => {
                 console.log('Error: Read image fail ' + error);
              });
          }
          products.push(p);
        });
        observer.next(products);
      });
    });
  }
```

4. Preview your app in the browser. You should now see the images.



Catalogue

Americano
$5.00

Cappuccino
$6.00

Latte
$6.00

Iced Mocha
$7.00

3 Read Product by Id from Database

In this task we are going to read a document by its `id` from **Firestore**.

1. Open *src > app > shared > services > firebase-product.service.ts*.

   Add a new `getProductById()` method.

firebase-product.service.ts

```
getProductById(id: string): Observable<any> {
   return new Observable((observer) => {
     this.productsRef.doc(id).get().then((doc) => {
       let data = doc.data();
        let p = new Product(data!['name'], data!['price'], data!['image'], doc!['id']);
         if (data!['category']) p.category = data!['category'];
         if (data!['vegetarian']) p.vegetarian = data!['vegetarian'];
         // If there's image, read from Firebase Storage
         if (data!['image']) {
           p.imagePath = data!['image'];
           const imageRef = firebase.storage().ref().child(data!['image']);
         imageRef.getDownloadURL()
           .then(url => {
             p.image = url;
             // Tell the subscriber that image is updated
             observer.next(p);
             console.log('Image is ' + p.image);
           }).catch(error => {
             console.log('Error: Read image fail ' + error);
           });
       }

       observer.next(p);
     });
   });
 }
```

2. Open **Edit Product Page** *src > app > edit-product > edit-product.page.ts*.
Change ***ProductService*** to ***FirebaseProductService***.
Use **Quick Fix** to import the class.

edit-product.page.ts

```
constructor(
    private route: ActivatedRoute,
    private router: Router,
    private productService: FirebaseProductService) {
```

3. We can’t use the old way to get product anymore from ProductService anymore. Delete or comment out the lines and instantiate a placeholder product.

edit-product.page.ts

```
constructor(
   private route: ActivatedRoute,
   private router: Router,
   private productService: FirebaseProductService) {

   // Practical 4 - Page Navigation
   this.productId = this.route.snapshot.params.id;

   // Practical 5 - Services
   ~~// this.product = this.productService.getProductById(this.productId);~~
   ~~// this.productImage = this.product.image;~~
   this.product = new Product('', 0, '');

   …
 }
```

4. Subscribe to the new service’s `getProductById()` method to get the data.

edit-product.page.ts

```
constructor(
   private route: ActivatedRoute,
   private router: Router,
   private productService: FirebaseProductService) {
   …

   // Practical 6 - Forms
    this.categories = ['Main', 'Beverage', 'Dessert'];
     this.editProductForm = new FormGroup({
      name: new FormControl(this.product.name, [Validators.required]),
      price: new FormControl(this.product.price, [EditProductPage.positiveNumber]),
      category: new FormControl(this.product.category),
      vegetarian: new FormControl(this.product.vegetarian)
   });

   // Practical 10 - CRUD
     this.productService.getProductById(this.productId)
       .subscribe(data => {
         this.product = data;
         if (this.product) {
           this.productImage = this.product.image;
 this.editProductForm.controls['name'].setValue(this.product.name);
 this.editProductForm.controls['price'].setValue(this.product.price);
this.editProductForm.controls['category'].setValue(this.product.category);

this.editProductForm.controls['vegetarian'].setValue(this.product.vegetari
an);
         }});
  }
```

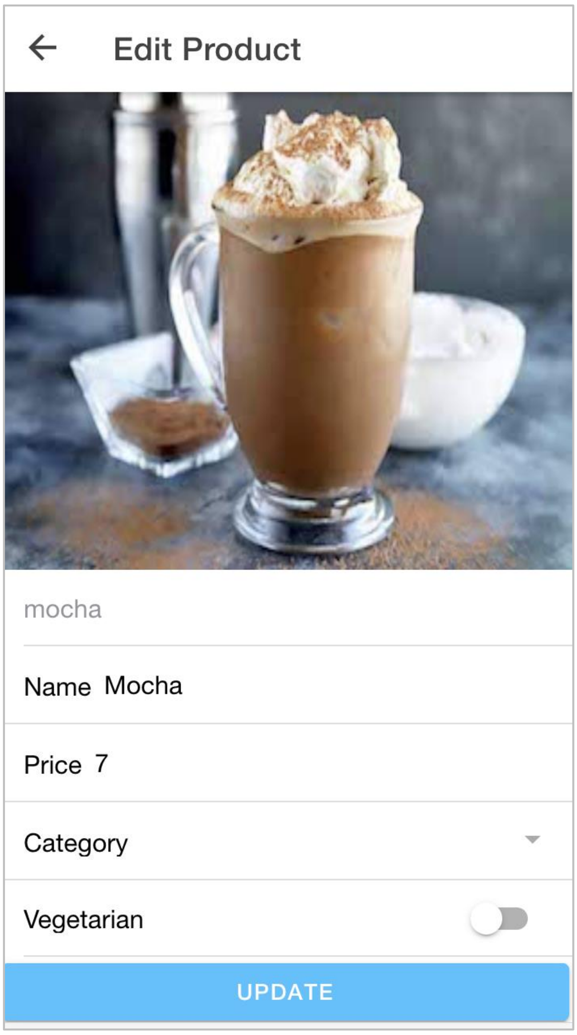
5. Comment off the following statement in update() function. Will fix it later

```
// this.productService.update(prod);
```

6. Test the app in your computer browser using `ionic serve`.

Go to **Edit Product Page**.

You should see the product retrieved from Database.



Edit Product

mocha

Name Mocha

Price 7

Category

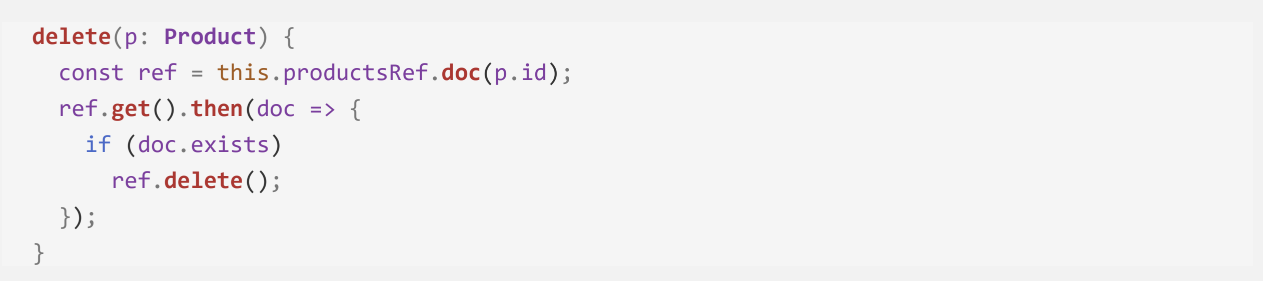
Vegetarian

4 Delete Product

1. Open *src > app > shared > models > firebase-product.service.ts*.

   Add `delete()` method.

firebase-product.service.ts



```
delete(p: Product) {
  const ref = this.productsRef.doc(p.id);
  ref.get().then(doc => {
    if (doc.exists)
      ref.delete();
  });
}
```

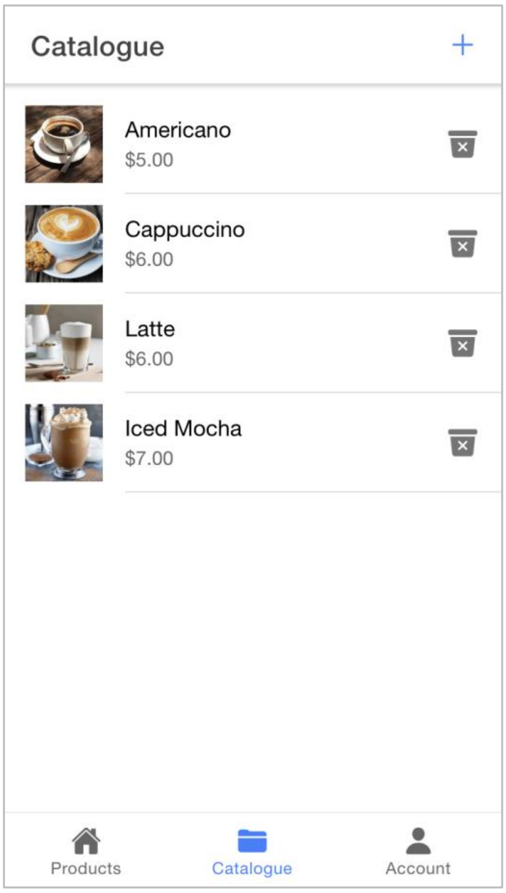
2. Goto tabs2.page.ts, uncomment the delete() function

```
delete(item: Product) {
    this.productService.delete(item);
  }
```

3. Preview your app in the browser.

   Go to **Catalogue Page**. Click on the **trash** icon to delete the item.

Verify in **Firebase console** that the item is deleted from the *'products'* collection.



5 Update Product

1. Open *src > app > shared > services > firebase-product.service.ts*.

   Add `update()` method.

firebase-product.service.ts

```
update(p: Product) {
  const ref = this.productsRef.doc(p.id);
  // Update compulsory fields. Do not update id and image
  ref.update({
    name: p.name,
    price: p.price,
  });
  // Update optional fields if not undefined
  if (p.category != undefined)
    ref.update({
      category: p.category
    });
  if (p.vegetarian != undefined)
    ref.update({
      vegetarian: p.vegetarian
    });
}
```

2. Open *src > app > edit-product > edit-product.page.ts*.

Uncomment the update() function

```
this.productService.update(prod);
```

3. Look at the `update()` method. You shouldn't have to modify anything. That's the benefit of using services.

The only important data is the `productId` which must correspond to the Firestore's document `id`.

edit-product.page.ts

```
update() {
  if (this.editProductForm.valid) {
    const prod = new Product(
      this.editProductForm.value.name,
      this.editProductForm.value.price,
     this.editProductForm.value.image,
      this.productId);
    prod.category = this.editProductForm.value.category;
    prod.vegetarian = this.editProductForm.value.vegetarian;
    this.productService.update(prod);

    this.router.navigate(['tabs/tab2']);
  }
}
```

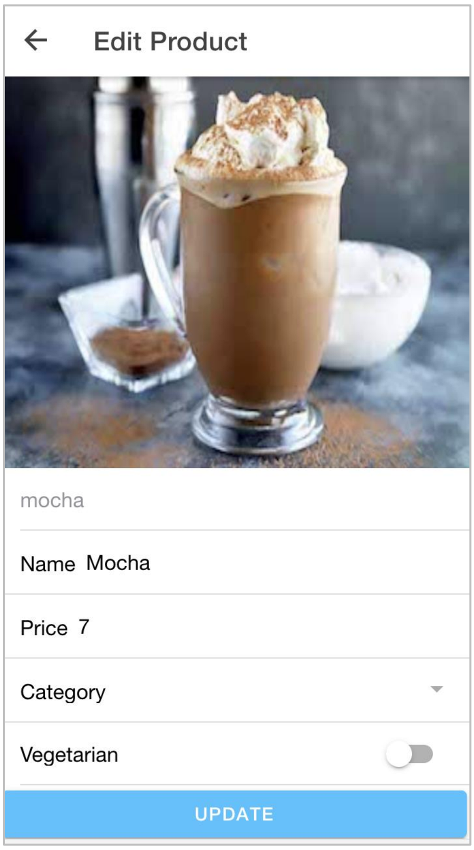
4. Preview your app in the browser using `ionic serve`.

Click on a list item to go to **Edit Product Page**.

Update the product data and click **Update**.

Verify that the changes appear on **Catalogue Page** too.

Verify the changes in **Firebase console**.



mocha

Name Mocha

Price 7

Category

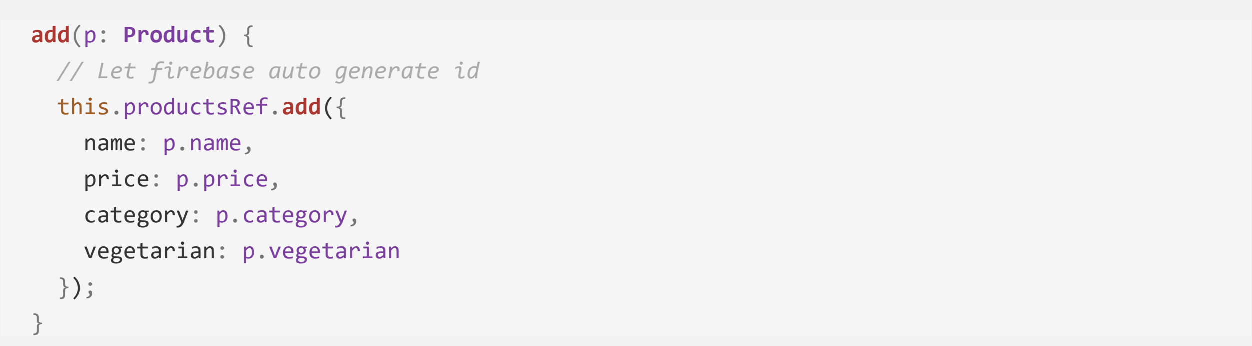
Vegetarian

6 Add Product

1. Open *src > app > shared > services > firebase-product.service.ts*.

   Add a new `add()` method.

firebase-product.service.ts



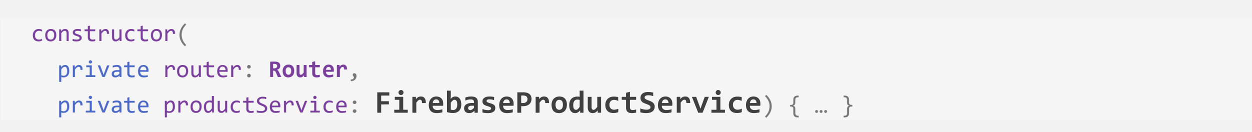
```
add(p: Product) {
  // Let firebase auto generate id
  this.productsRef.add({
    name: p.name,
    price: p.price,
    category: p.category,
    vegetarian: p.vegetarian
  });
}
```

2. Open *src > app > add-product > add-product-page.ts*.

   Inject ***FirebaseProductService***.

   Use **Quick Fix** to import the class.

add-product-page.ts

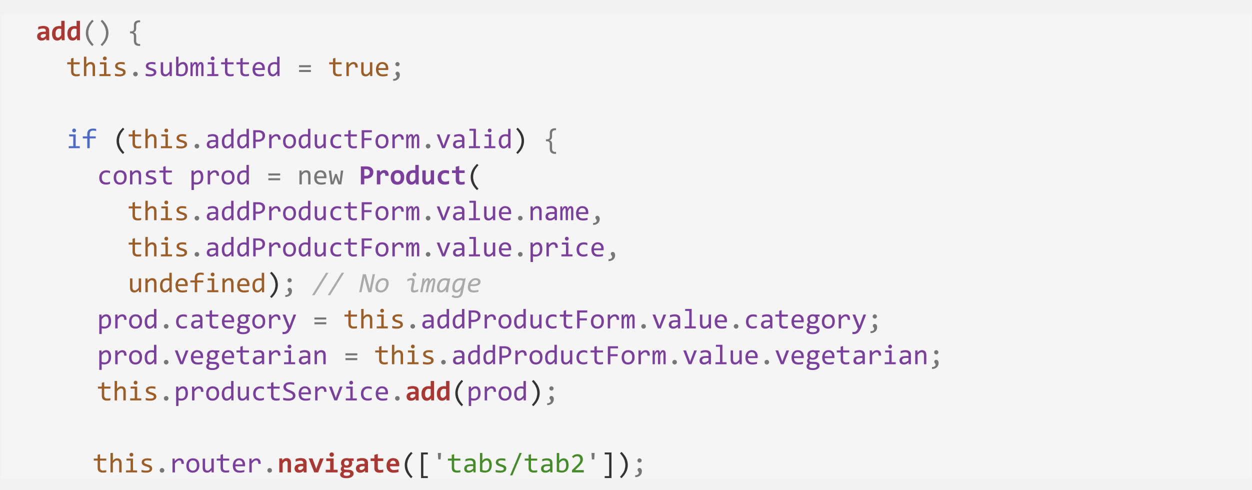


```
constructor(
  private router: Router,
  private productService: FirebaseProductService) { … }
```

3. In **Add Product Page**, look at the `add()` method. you shouldn’t have to modify anything.

   The product `id` is not a concern as it will be automatically generated by **Firestore**.

add-product-page.ts

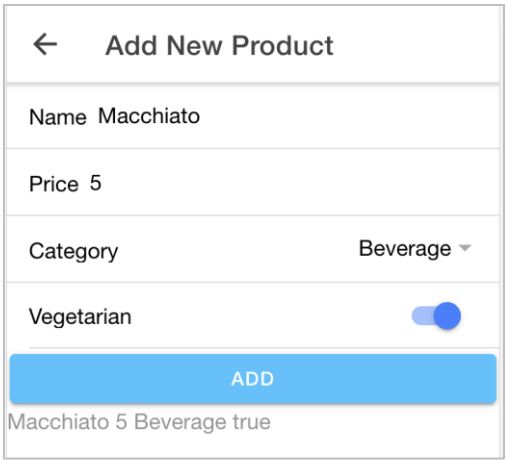


```
add() {
  this.submitted = true;

  if (this.addProductForm.valid) {
    const prod = new Product(
      this.addProductForm.value.name,
      this.addProductForm.value.price,
      undefined); // No image
    prod.category = this.addProductForm.value.category;
    prod.vegetarian = this.addProductForm.value.vegetarian;
    this.productService.add(prod);

    this.router.navigate(['tabs/tab2']);
    }
```

4. Preview your app in the browser.

   Go to **Add Product Page**.

   Fill in the data and click **Add** button.



Add New Product

Name Macchiato

Price 5

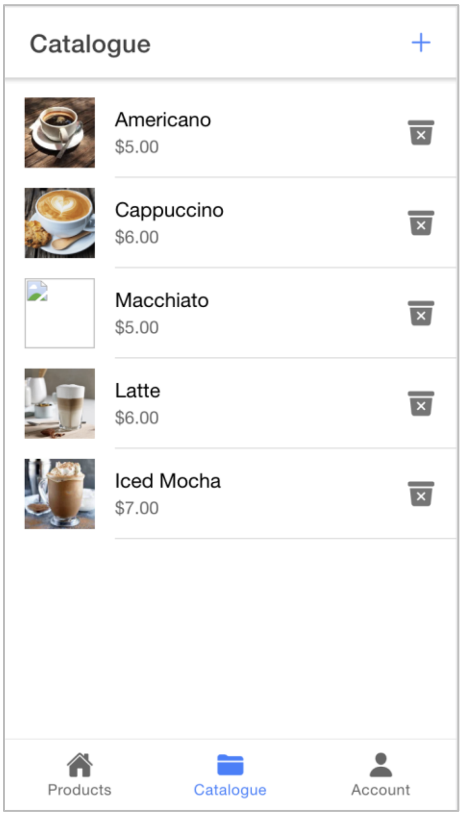
Category Beverage

Vegetarian

ADD

Macchiato 5 Beverage true

You should see the new product in the **Catalogue Page**.



Catalogue

Americano
$5.00

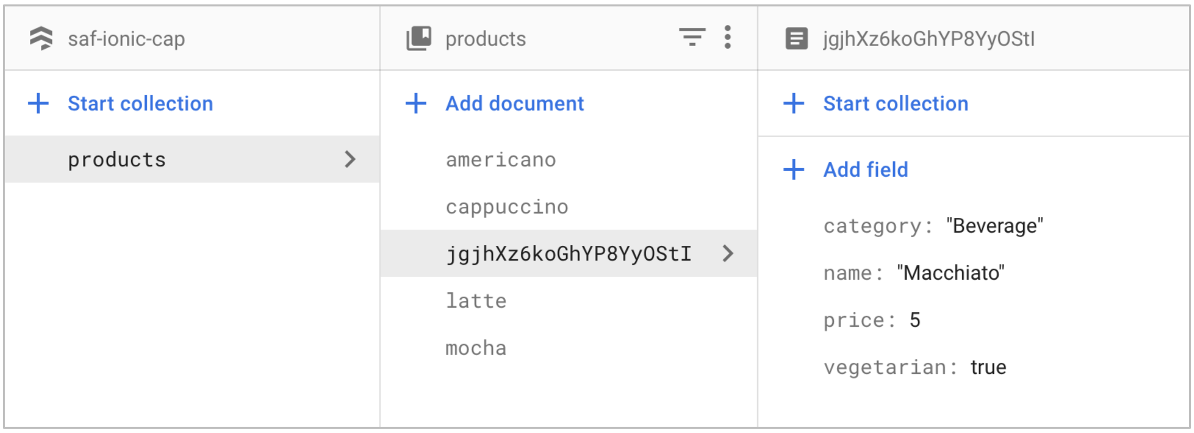
Cappuccino
$6.00

Macchiato
$5.00

Latte
$6.00

Iced Mocha
$7.00

Go to **Firebase console**. Verify that the new product is created with an auto-generated id. Auto-generated id help us ensure that each document has a unique id in the 'products' collection.



saf-ionic-cap

+ Start collection

products

products

+ Add document

americano

cappuccino

jgjhXz6koGhYP8YyOStI

latte

mocha

jgjhXz6koGhYP8YyOStI

+ Start collection

+ Add field

category: "Beverage"

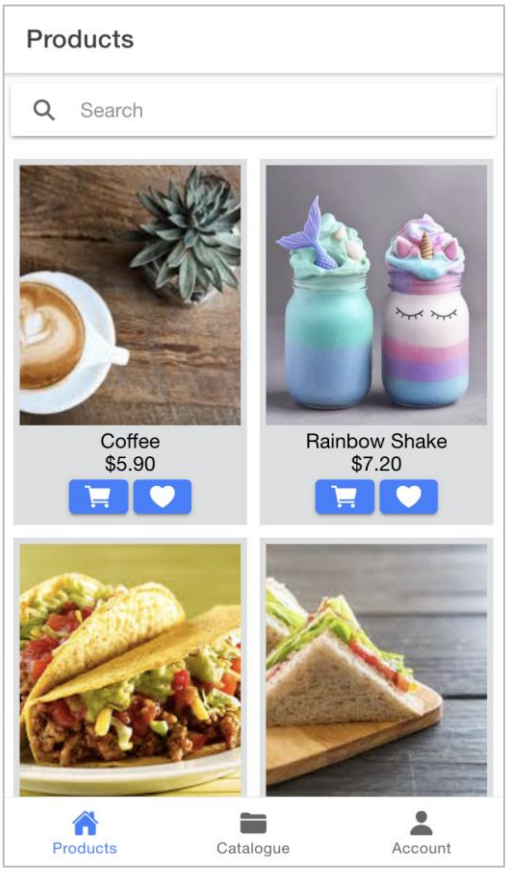
name: "Macchiato"

price: 5

vegetarian: true

7 Add Cart and Cart items

(Challenge) Are you able to add to the *'cart'* in **Firestore** when the user clicks on the cart button in the **Products Page**?



~ End ~