GatorMart Documentation

April 1st, 2022

Sprint 3

Team Members:

Front End

1. Nitin Ramesh

2. Bhanu Prakash Reddy Palagati

Back End

1. Gowtham Reddy Eda

2. Vamsi Krishna Reddy Komareddy

Introduction:

The GatorMart application is an online market build with Angular and Go language. Built from the ground up, the application aims to seamlessly connect buyers to their products and sellers to their target audience. Its standout features include allowing the users to select a target audience while buying or selling. This allows to buy products or sell products in single or wholesale quantities while specifying a target audience such as students, professionals, farmers, etc.

Tech Stack:

Frontend: Angular 11.0 with TypeScript
Frontend Test cases: Cypress and Jest Framework
Backend: Go Language
Database: MySQL

Frontend:

The front end of the application is built using Angular 11. Angular is a framework rather than a package that provides all the essential functional requirements out of the box. This will force the developers to follow a pattern, as a result, we have fewer decisions to take for the organization and spend time more on what matters.

We have used material design and there is a package named Angular Materials which helps us to implement the material style components easily and efficiently.

Test cases have been written using Jest for unit testing and Cypress for integration testing, this has ensured the smooth integrated working of the frontend and backend.

The application demo consists of the following pages: Login page, Registration page, "List view" with product filters, create product page and the "Detailed view" with google maps integration:

1) Registration Page:
   The new user registers for the GatorMart account at this page. The user enters the details and the details are validated before the user account Is created. The user identifier is the email they register by and the password is minimum 8 characters with atleast 1 capital letter, 1 number and 1 symbol.

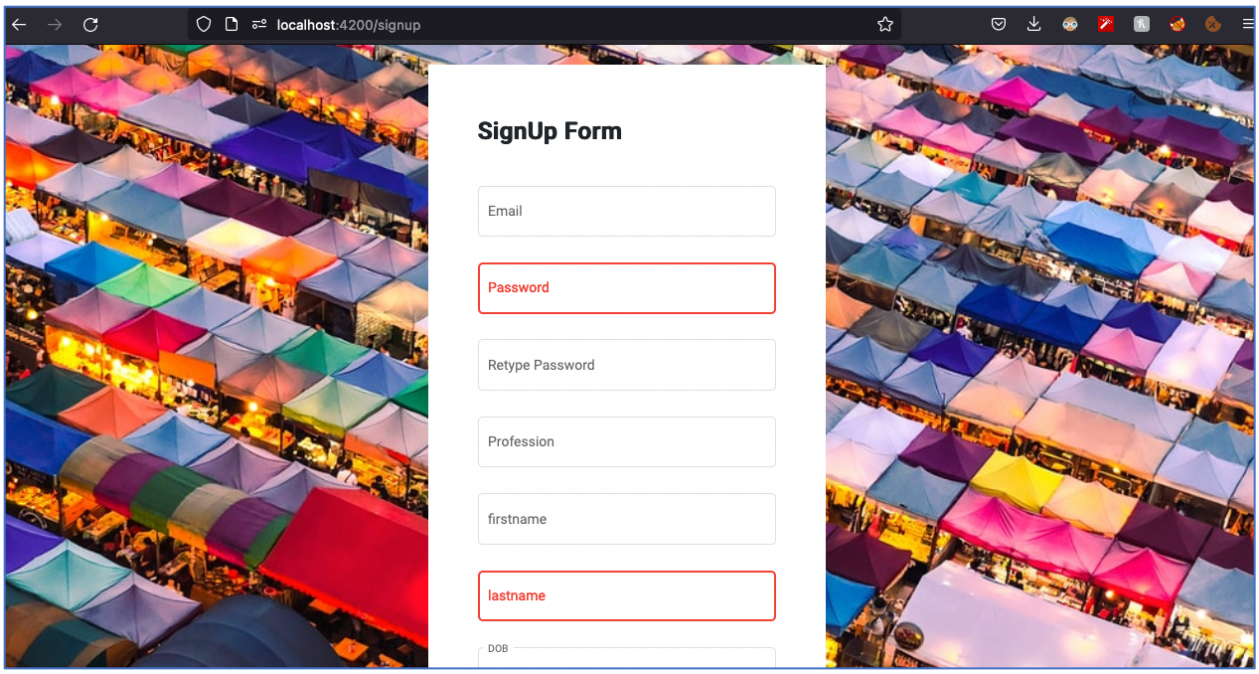


Figure 1: Registration Page

2) Login Page:

The traditional login of username and password is presented to the user upon visiting the GatorMart application. The username and password is taken and redirected to the main homepage of the application.

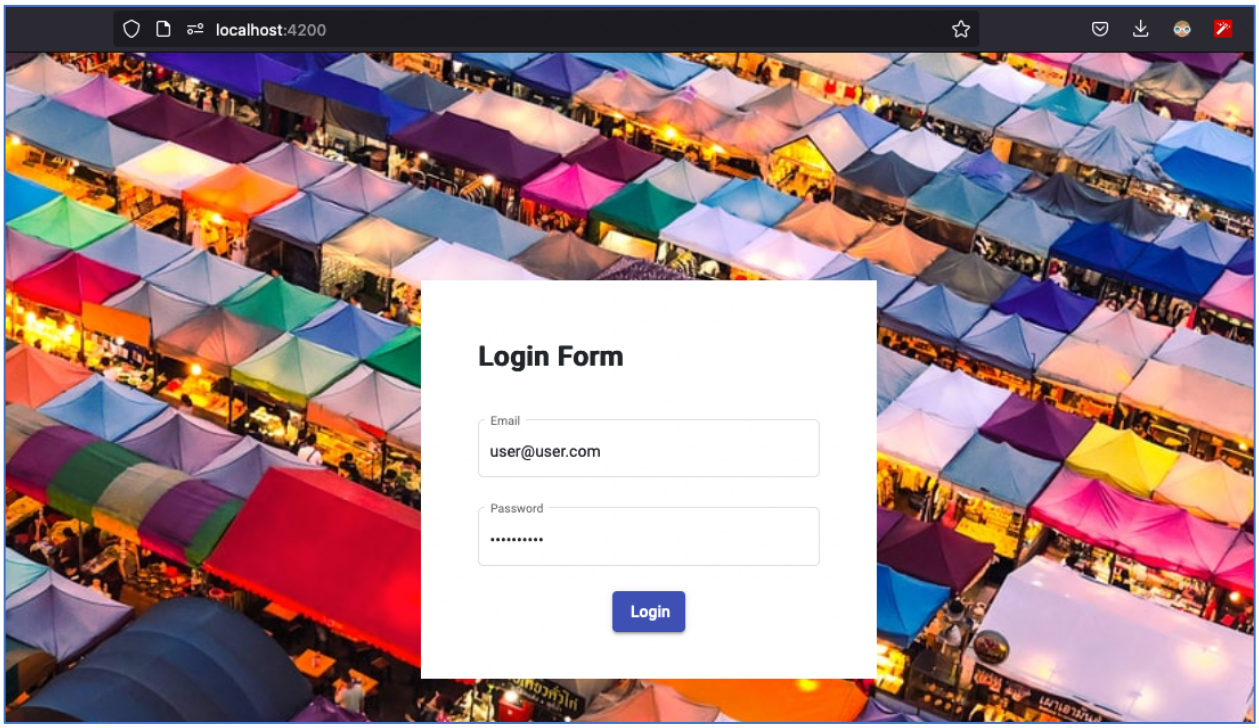


Figure 2: Login Page

3) List View:

The List view is the main screen that greets the user when opening the GatorMart application. It is a block view of all the products, that have been hand-picked by our algorithms to suit the user's taste.

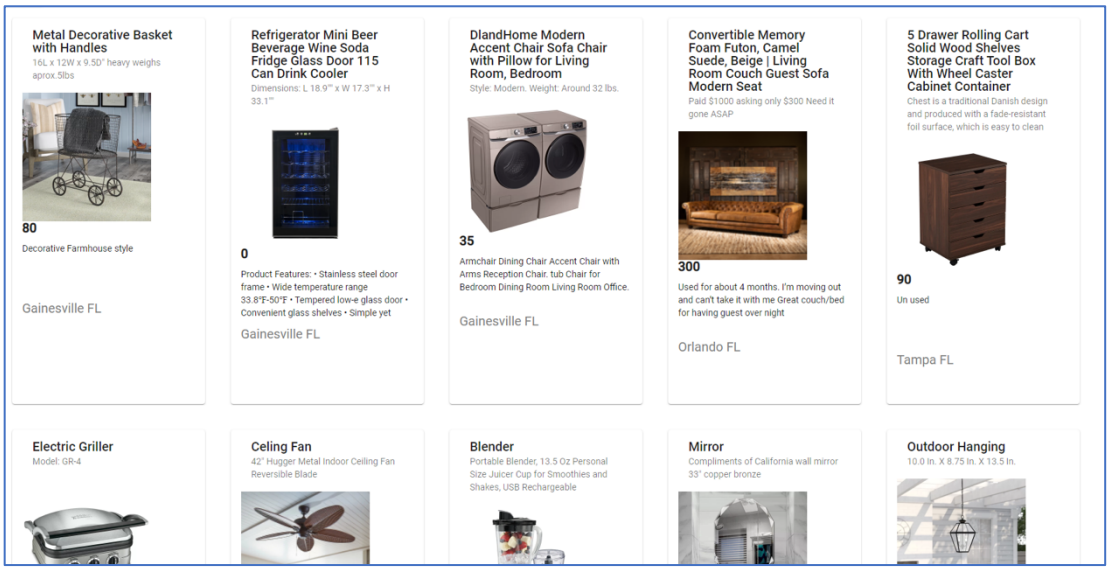


Image: List View

The products are displayed here in a small window shaped material design as shown below:

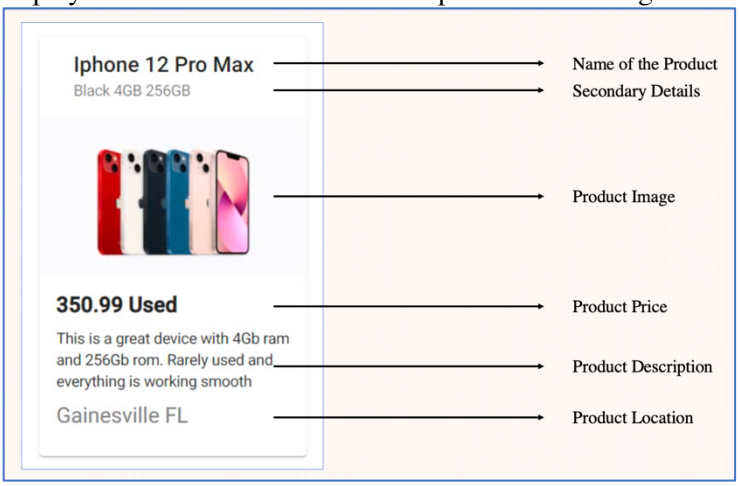


Image: Product Details

This method favors readability, as the most important product information is conveyed to the user directly as they scroll the different listings on the site.

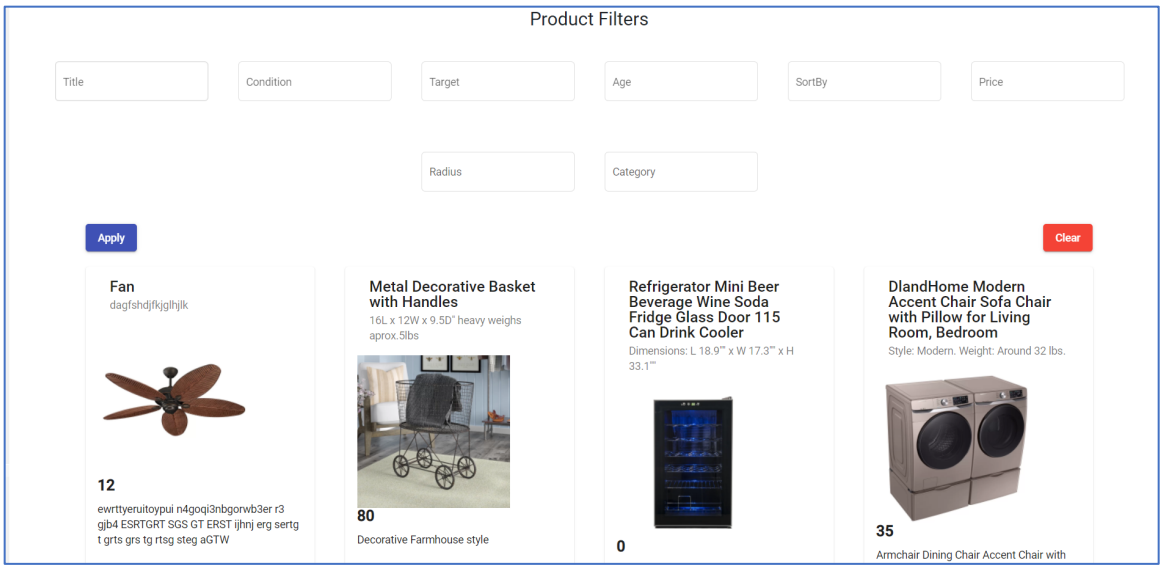


Image: Product Filters

The List view function also contains the product filters, with options such as title,condition, target, age etc. This allows the users to segregate the products and only list those that are relevant to the search parameters.

4) Detailed View:

The Detailed View is presented when the user clicks on a product in the List view. This view, as the name suggests, gives more detailed information regarding the selected product such as:

- A carousel of images of the product
- Age of the product
- Detailed description
- Price etc.
- Option to Edit/Remove the Add
- Map indicating the location of the seller

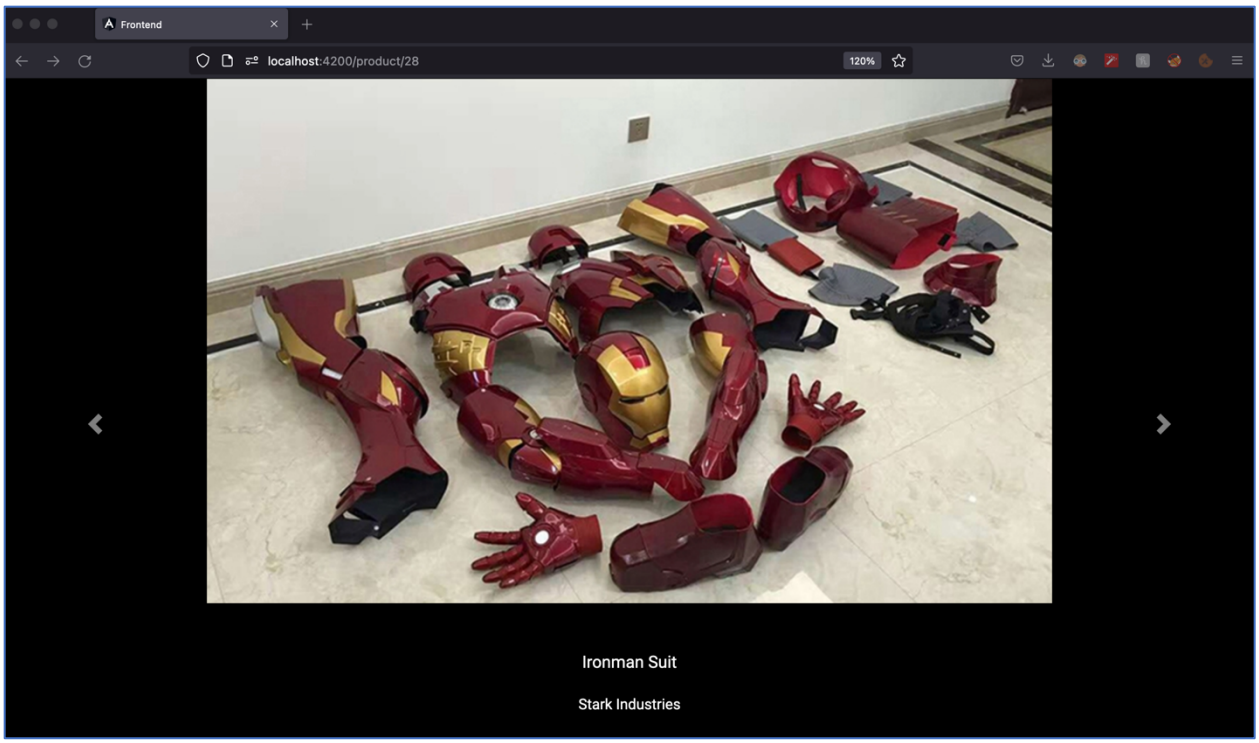


Image: Image carousel

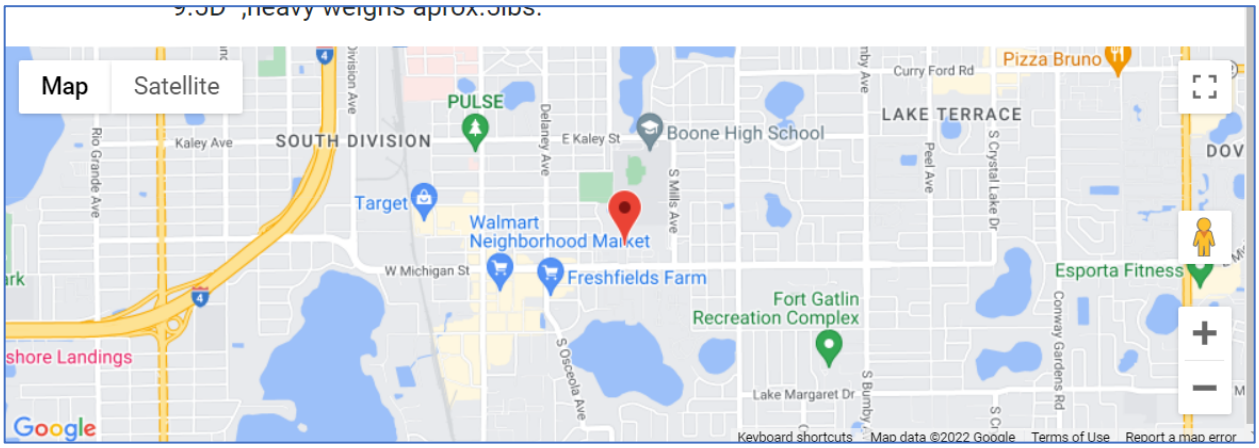


Image: Detailed View

All details are conveniently placed in one area with multiple images to give a better overall understanding of the product.

Edit function : The product details can be edited by the users via the edit button, where the updated details is entered and reflected in the site.

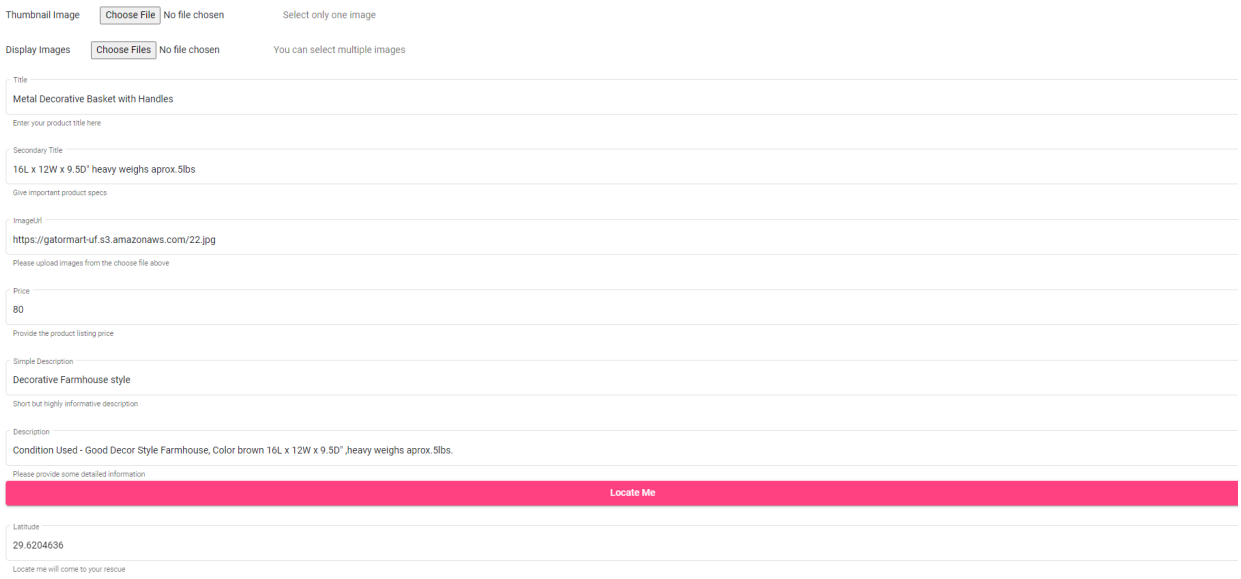


Image: Edit View

Delete function : The product can be deleted in the product details view. This will verify the deletion function with the user and proceed to delete the product.

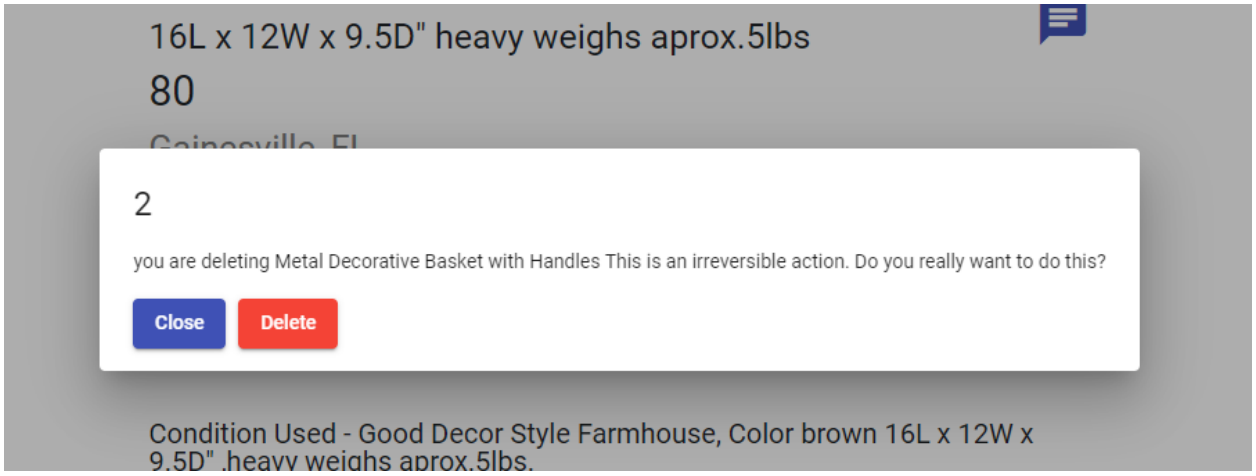


Image: Delete Ad

5) <u>Create Product View</u>:
The create product view confirses of a form where the user can add all relevant details and set the product up for listing. This includes the main image, carousel images and device specific data.

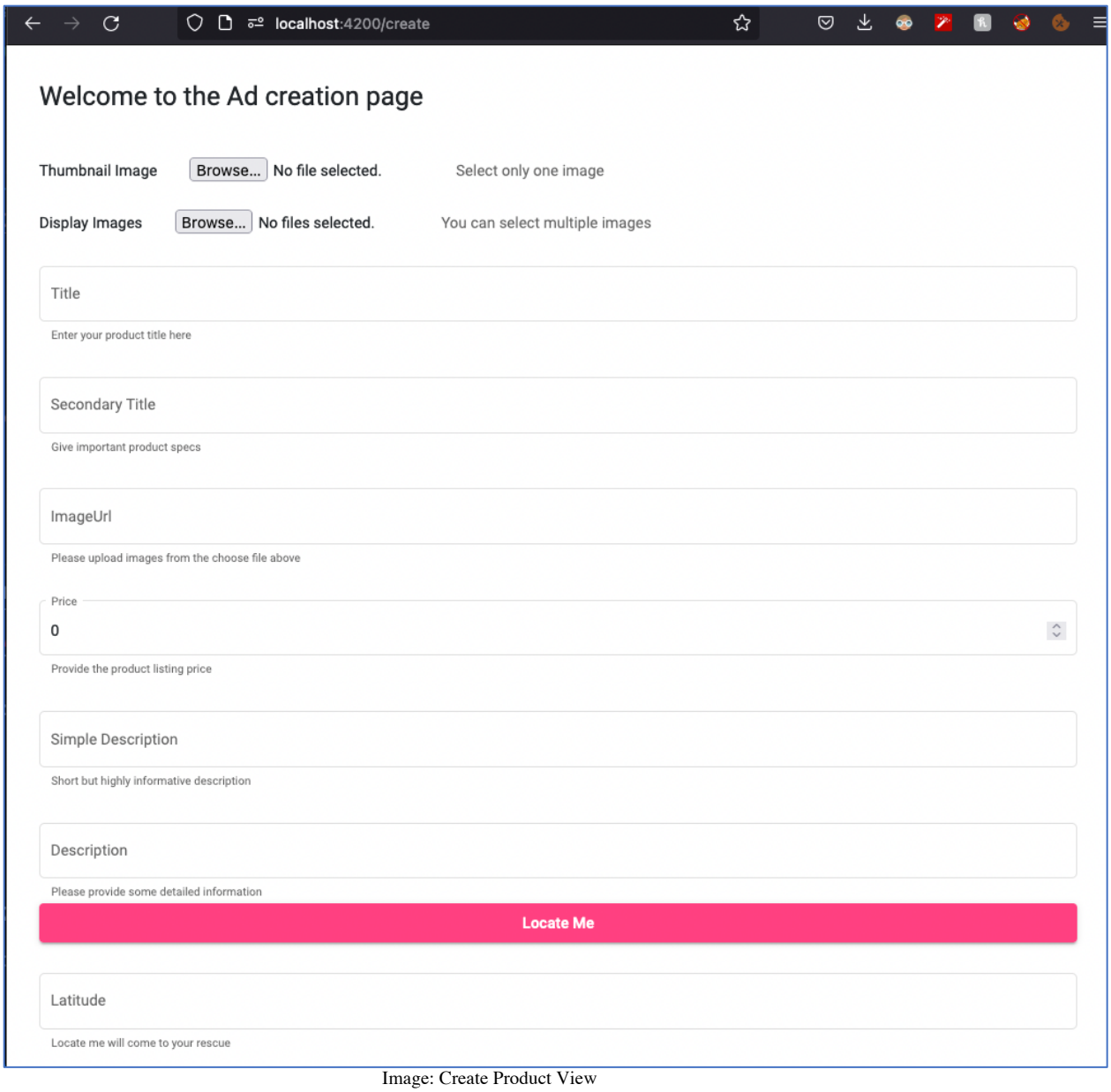
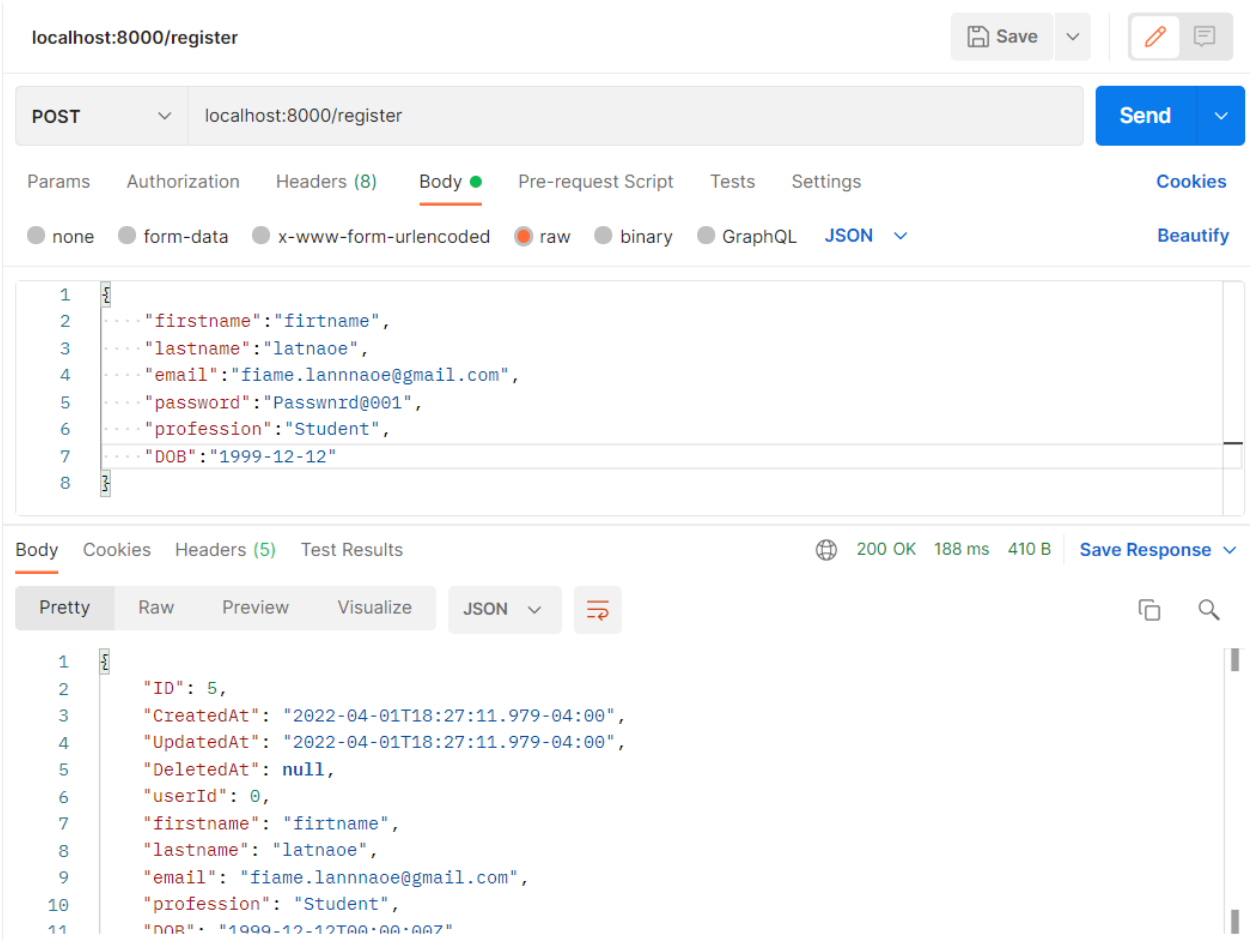


Image: Create Product View

Backend:

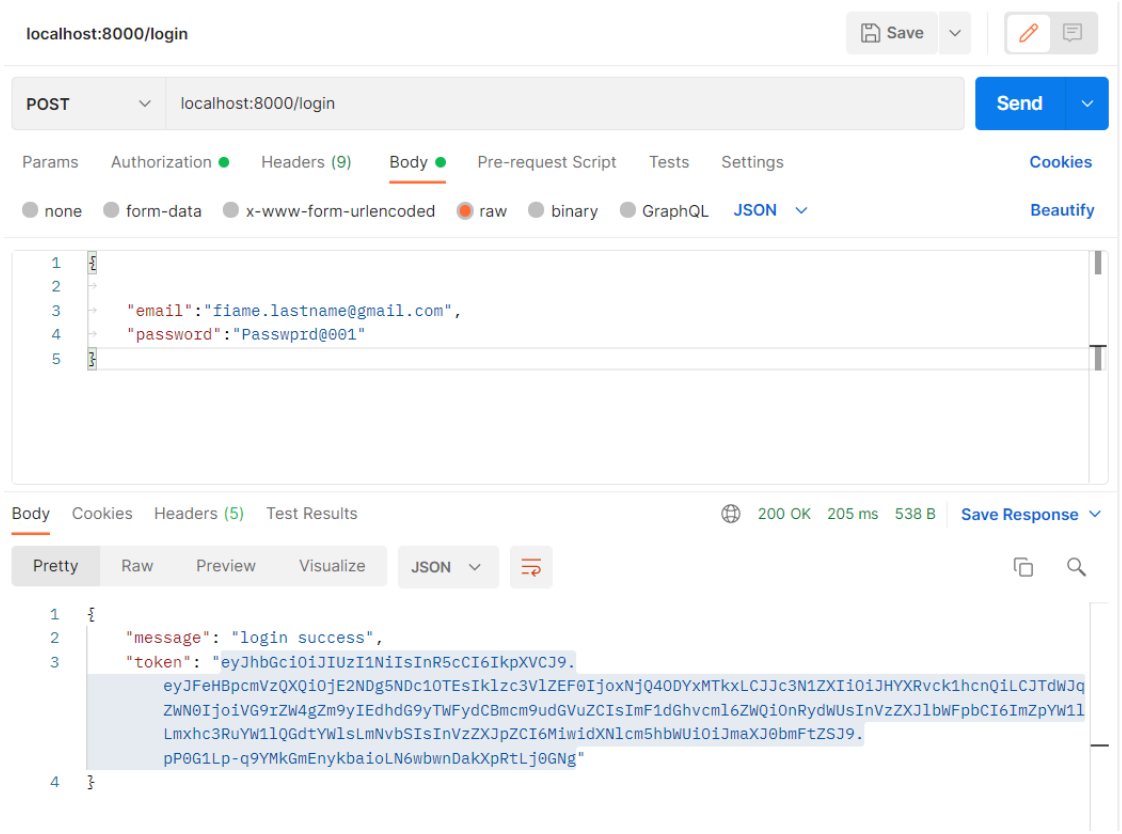
1) Register:

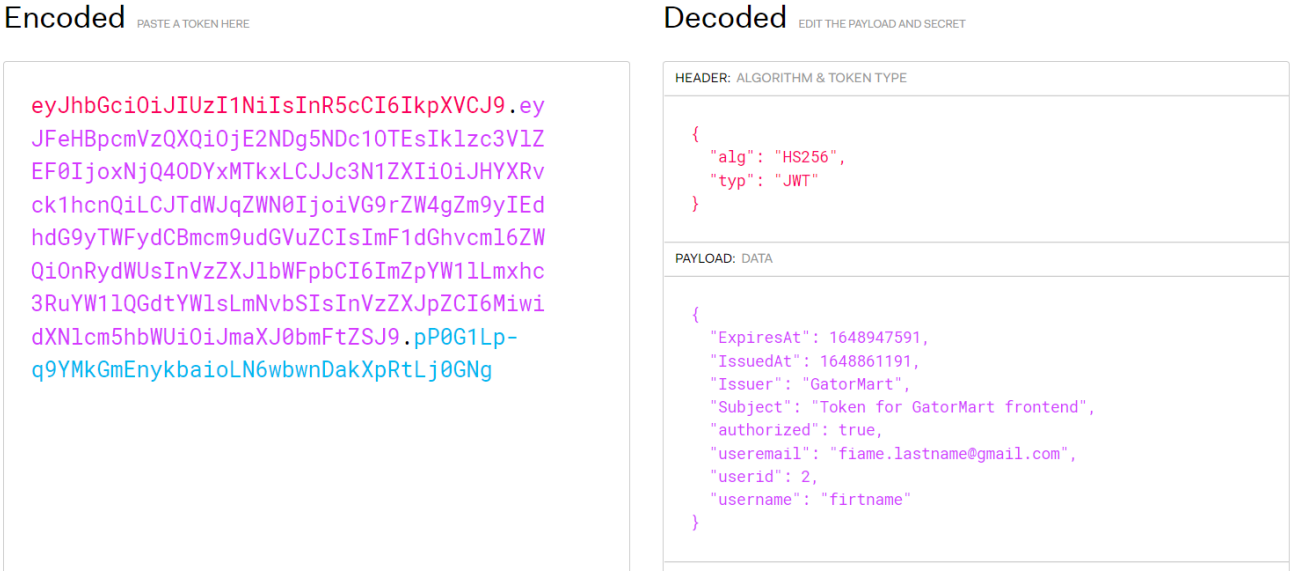
Added new fields and updated the struct of golang, some fields firstname,lastname, email, Password, Proffesion and DateofBirth which helps us to focous on the targeted audienece for selling and buying of products.



2) Login:

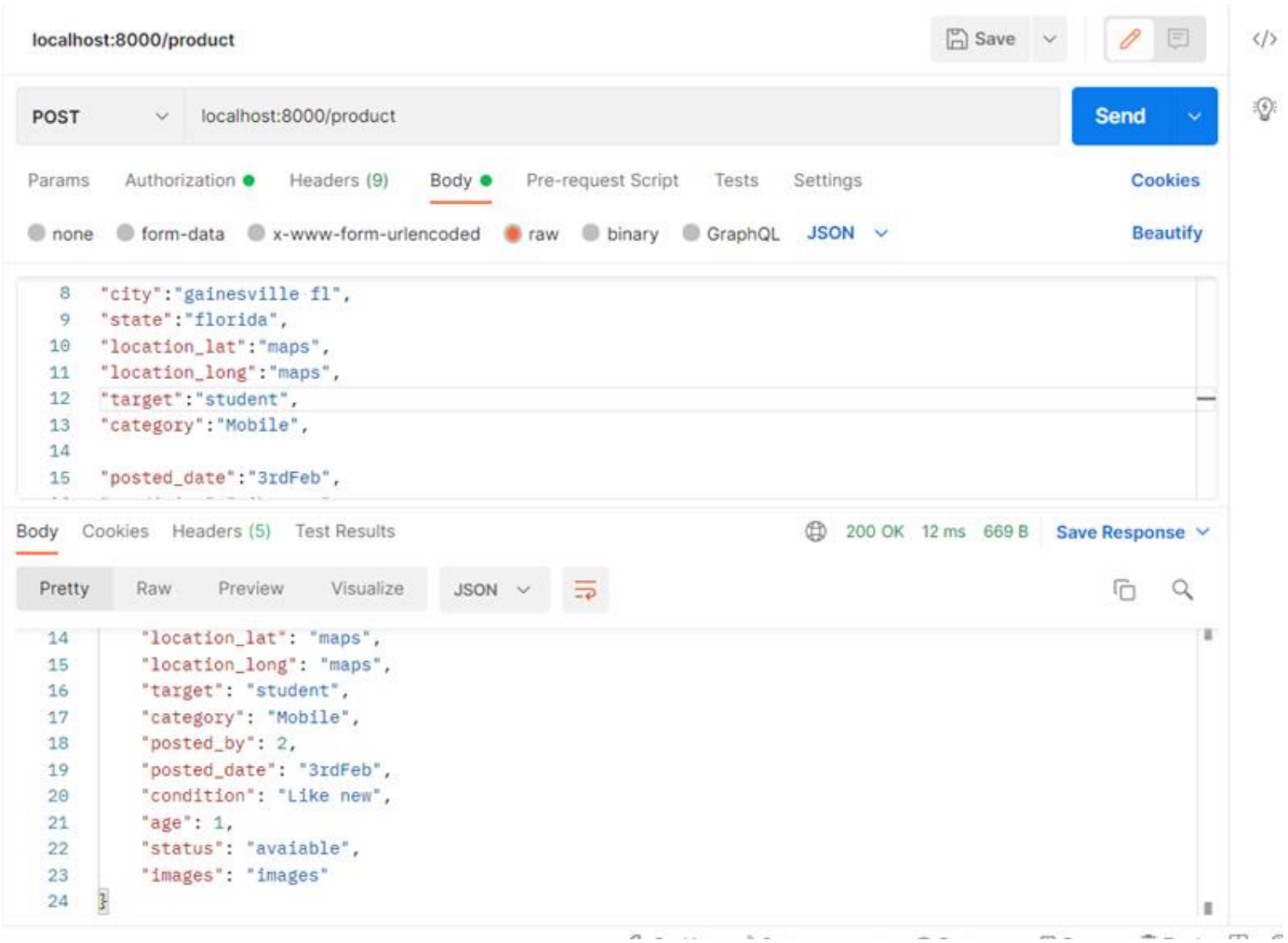
Upadted login end point to generate JWT token based on userid, useremial and username.





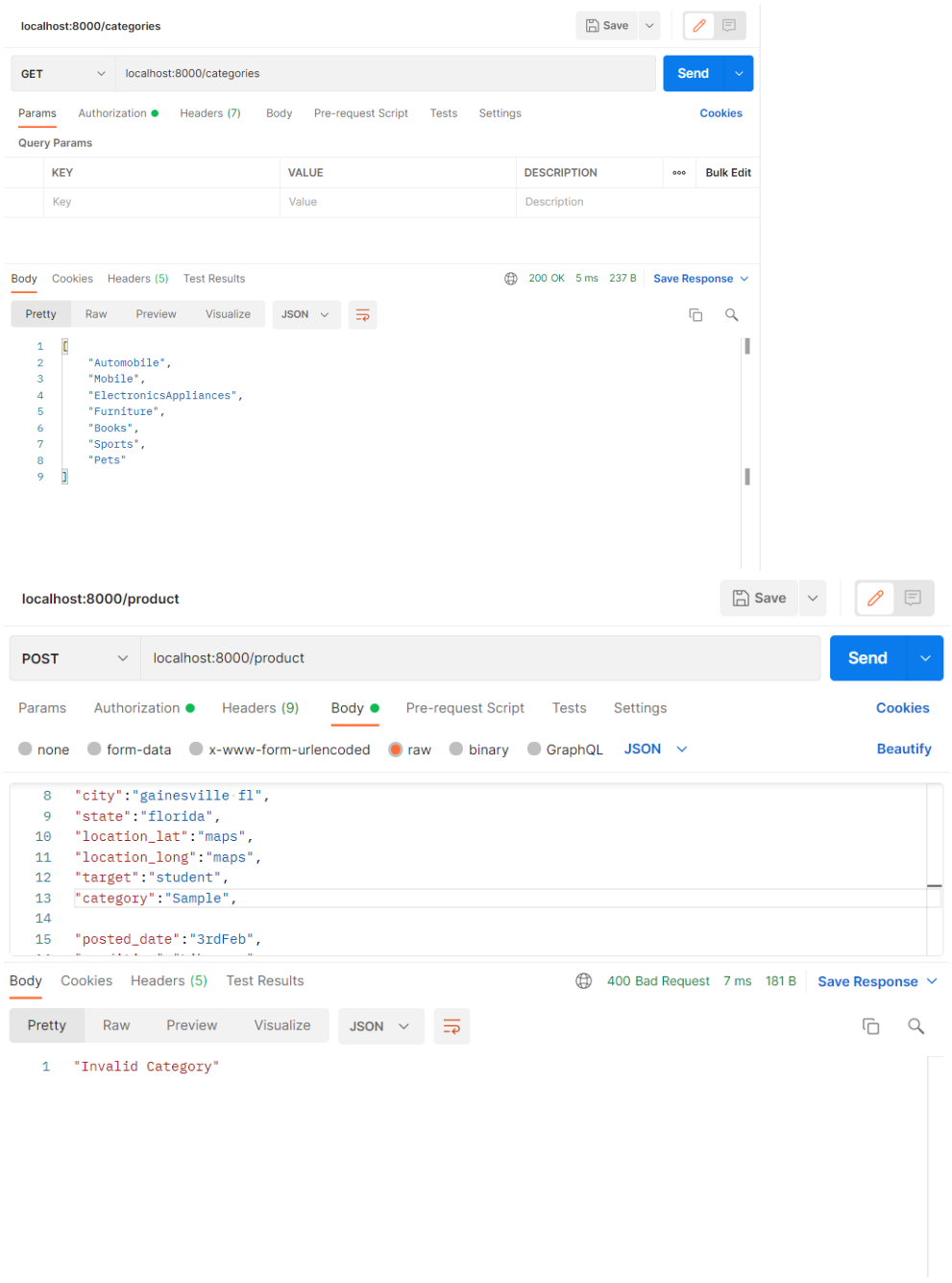
3) <u>Productsave:</u>

Updated save method to automatically update postedby field by taking user details from JWT token.



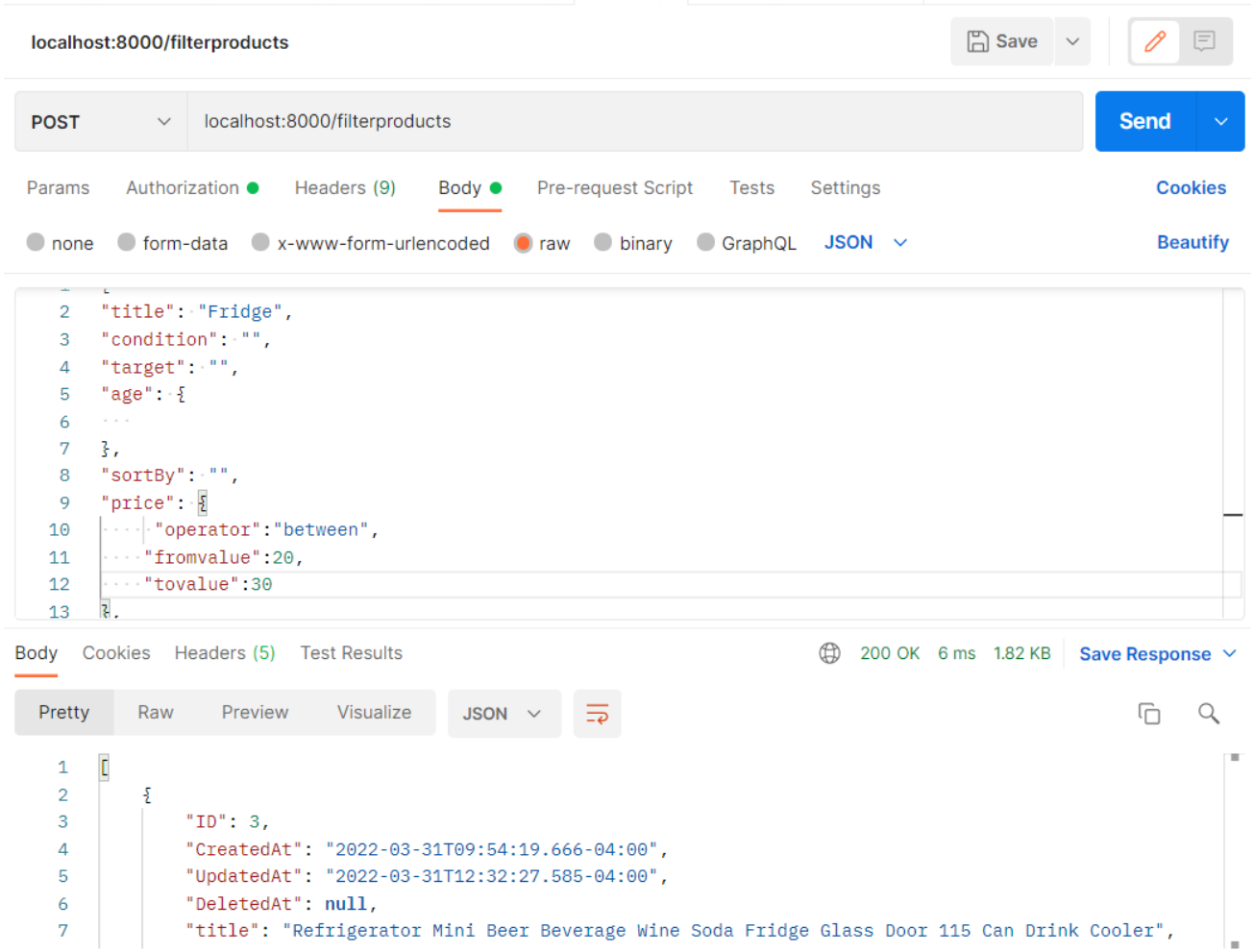
4) <u>Category Dropdown:</u>

Added categories dropdown where users can select the category type only from the dropdown. Updated save product and update product end point to restrict the user selection of categoris from the dropdown. It throws an error if the value doesnot present in the dropdown.



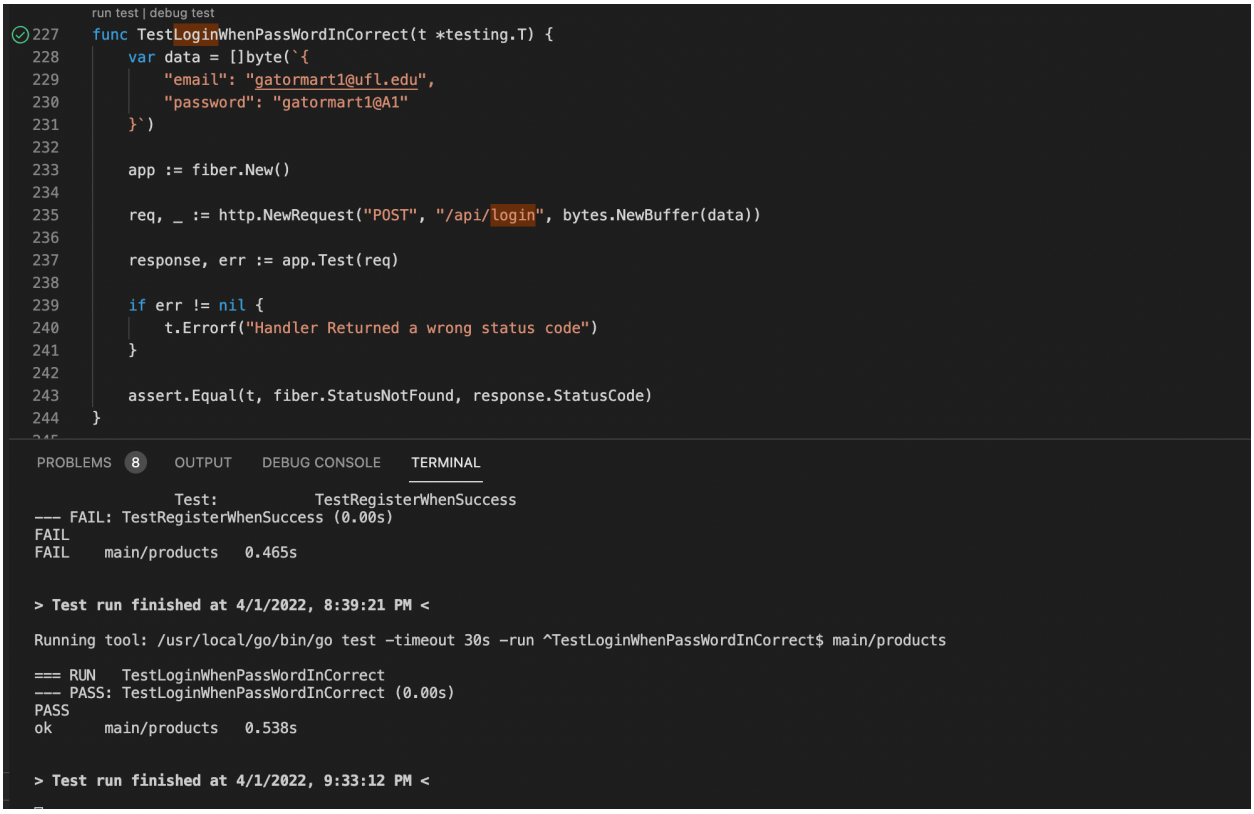
5) Filter Products:

User can filter products based on partial title match, target, category and price based upon operater, fromvalue, tovalue.



6) Unit test Cases:

Implemented unit test cases has been implemented for all the functions.



```
run test | debug test
227  func TestLoginWhenPassWordInCorrect(t *testing.T) {
228      var data = []byte(`{
229          "email": "gatormart1@ufl.edu",
230          "password": "gatormart1@A1"
231      }`)
232
233      app := fiber.New()
234
235      req, _ := http.NewRequest("POST", "/api/login", bytes.NewBuffer(data))
236
237      response, err := app.Test(req)
238
239      if err != nil {
240          t.Errorf("Handler Returned a wrong status code")
241      }
242
243      assert.Equal(t, fiber.StatusNotFound, response.StatusCode)
244  }
```

```
PROBLEMS 8   OUTPUT   DEBUG CONSOLE   TERMINAL

                Test:           TestRegisterWhenSuccess
--- FAIL: TestRegisterWhenSuccess (0.00s)
FAIL
FAIL    main/products   0.465s


> Test run finished at 4/1/2022, 8:39:21 PM <

Running tool: /usr/local/go/bin/go test -timeout 30s -run ^TestLoginWhenPassWordInCorrect$ main/products

=== RUN   TestLoginWhenPassWordInCorrect
--- PASS: TestLoginWhenPassWordInCorrect (0.00s)
PASS
ok      main/products   0.538s


> Test run finished at 4/1/2022, 9:33:12 PM <
```