

GatorMart Documentation

April 20st, 2022

Sprint 4

Team Members:

Front End

Nitin Ramesh
Bhanu Prakash Reddy Palagati

Back End

Gowtham Reddy Eda
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

Hosted URL:

The application is hosted at the following URL: <https://gatormartuf.web.app/>

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.

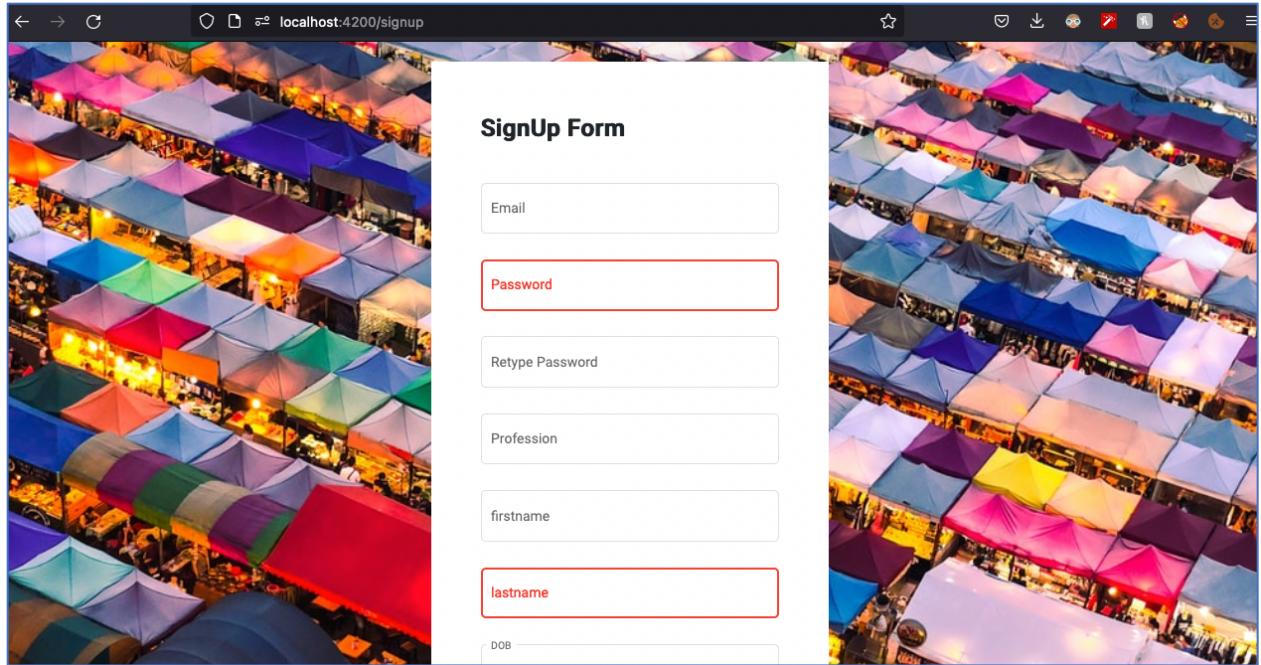


Figure 1: Registration Page

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.

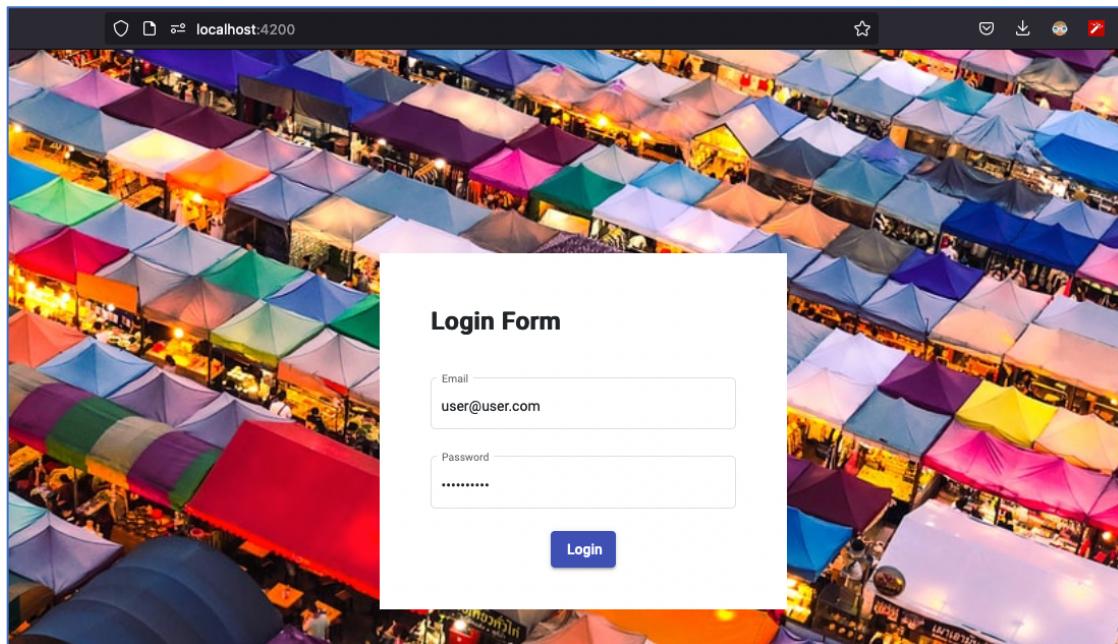


Figure 2: Login Page

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.

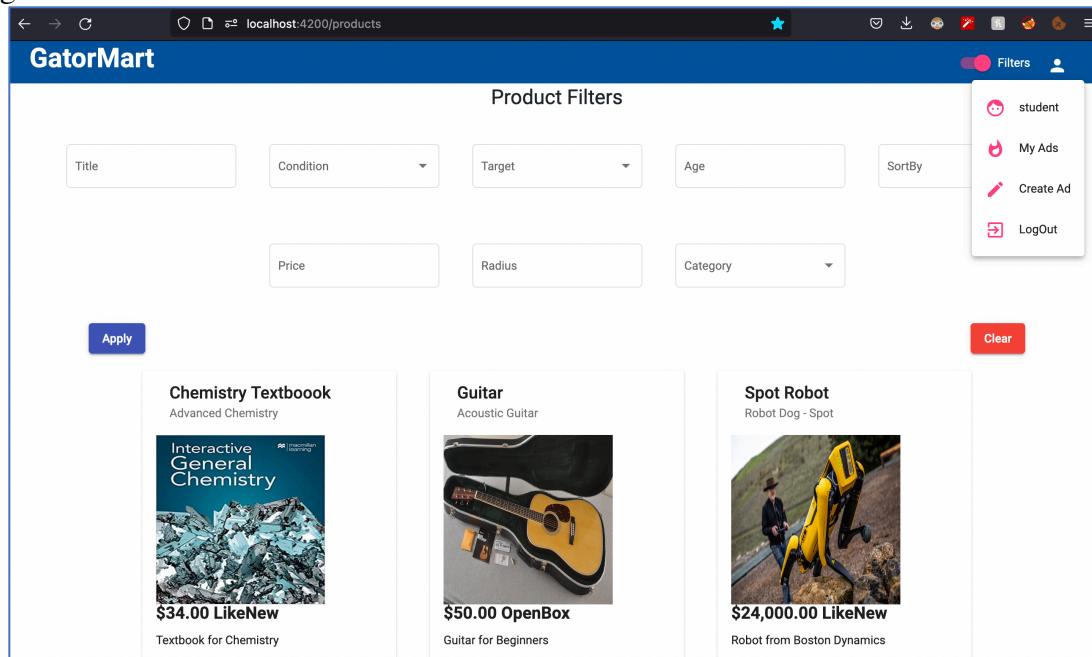


Image: List View

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.

The screenshot shows the GatorMart website interface. At the top, there is a blue header bar with the text "GatorMart" on the left and a "Filters" button with a person icon on the right. Below the header is a section titled "Product Filters" containing several input fields: "Title", "Condition" (with a dropdown arrow), "Target" (with a dropdown arrow), "Age", "SortBy" (with a dropdown arrow), and "Price". Further down are two more dropdown fields: "Radius" and "Category". In the center, there is a large "Apply" button in a blue box and a "Clear" button in a red box. Below these buttons are four product cards displayed in a grid:

- Chemistry Textbook**
Advanced Chemistry

\$34.00 LikeNew
Textbook for Chemistry
- Guitar**
Acoustic Guitar

\$50.00 OpenBox
Guitar for Beginners
- Spot Robot**
Robot Dog - Spot

\$24,000.00 LikeNew
Robot from Boston Dynamics
- DSLR Camera**
Camera with lens kit

\$630.00 New
Professional Camera for sale

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

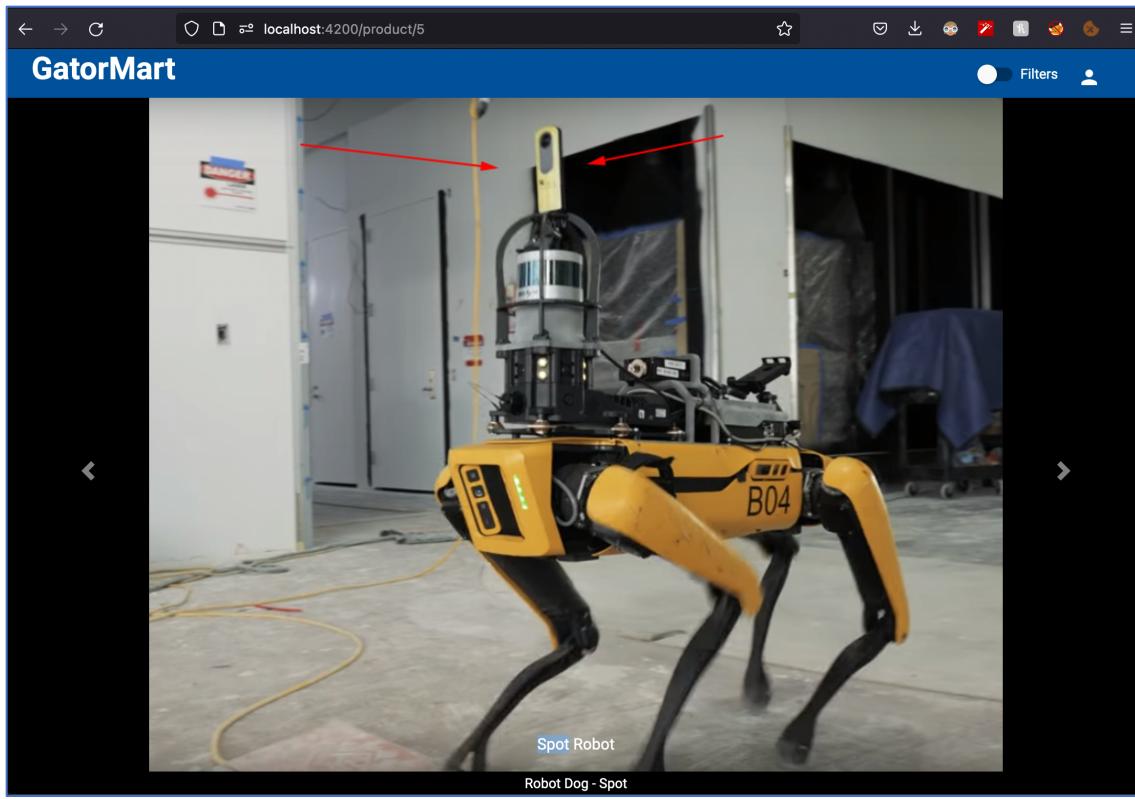


Image: Image carousel

Spot Robot
Robot Dog - Spot
24000
Gainesville, Florida

Product Age:- 1
LikeNew

High precision robot dog from boston dynamics, useful for scientific and research purposes.

Map Satellite

Orlando Health H
Arnold Palmer Hospital...
PULSE
Target
Walmart Neighborhood Market
Freshfields Farm
Fort Gatlin Recreation Complex
Curry Ford Rd
S Mills Ave
E Michigan St
Lake Margaret Dr
S Gandy Ave
SC
Curry Ford Rd
Sedano
DIX
Publix Super Market at Conway Plaza...
Dover Estates
Bryn Mawr
Lake Margaret Dr
Conway Gardens Rd
SC

Keyboard shortcuts Map data ©2022 Google Terms of Use Report a map error

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.

Thumbnail Image No file chosen Select only one image

Display Images No file chosen You can select multiple images

Title
Metal Decorative Basket with Handles
Enter your product title here

Secondary Title
16L x 12W x 9.5D' heavy weighs approx.5lbs
Give important product specs

ImageURL
https://gatormart-uf.s3.amazonaws.com/22.jpg
Please upload images from the choose file above

Price
80
Provide the product listing price

Simple Description
Decorative Farmhouse style
Short but highly informative description

Description
Condition Used - Good Decor Style Farmhouse, Color brown 16L x 12W x 9.5D'.heavy weighs approx.5lbs.
Please provide some detailed information

Latitude
29.6204636
Locate me will come to your rescue

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.

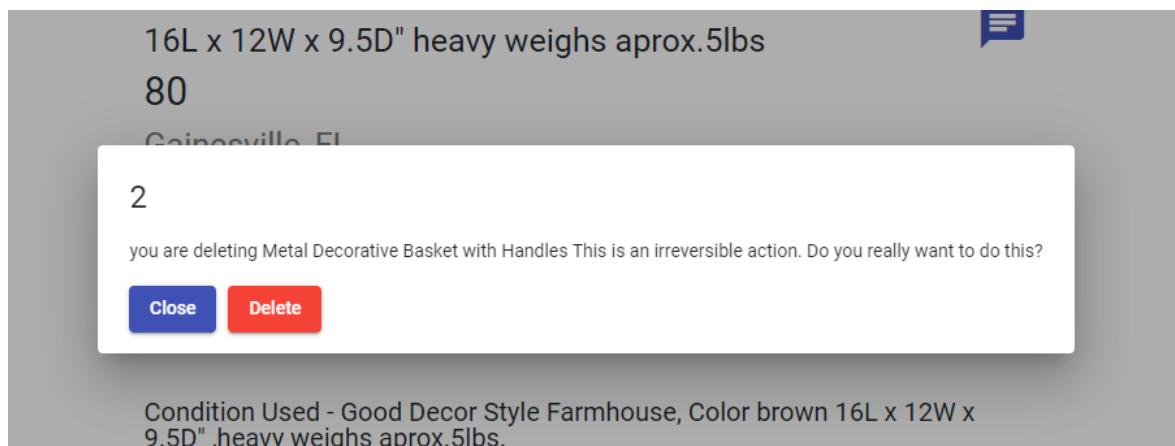


Image: Delete Ad

5) Create Product View:

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.

localhost:4200/create

Welcome to the Ad creation page

Thumbnail Image No file selected. Select only one image

Display Images No files selected. You can select multiple images

Title
Enter your product title here

Secondary Title
Give important product specs

ImageUrl
Please upload images from the choose file above

Price
0

Provide the product listing price

Simple Description
Short but highly informative description

Description
Please provide some detailed information

Latitude
Locate me will come to your rescue

Image: Create Product View

6) My Ads View:

The view displays the ads that the user has posted and allows to edit and delete the ads that have been posted.

The screenshot shows a web browser window for 'GatorMart' at 'localhost:4200/myads'. The title bar says 'GatorMart' and the address bar says 'localhost:4200/myads'. The main content area is titled 'MyAds'. It displays three ads in a grid:

- Spot Robot**
Robot Dog - Spot

\$24,000.00 LikeNew
Robot from Boston Dynamics
Gainesville Florida
- DSLR Camera**
Camera with lens kit

\$630.00 New
Professional Camera for sale
Gainesville Florida
- Cypress Electricals**
Cypress Progress

\$999.00 Used
Satellite downlink in progress
Gainesville Florida

Image: Ads Page

FrontEnd Demo Video:

https://drive.google.com/file/d/1OIYDkWaMdK3yeMaRX70MPOkC_8XVNvem/view?usp=sharing

Cypress Video:

<https://drive.google.com/file/d/1Suqm1y8tGllgZqCPowNzBWqs7YcZyIh8/view?usp=sharing>

Backend:

1) Register:

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

The screenshot shows a Postman interface with the following details:

- URL: `localhost:8000/register`
- Method: `POST`
- Headers: `(8)`
- Body (JSON):

```
1
2   "firstname": "firtnname",
3   "lastname": "latnaoe",
4   "email": "fiaime.lannnaoe@gmail.com",
5   "password": "Passwnrd@001",
6   "profession": "Student",
7   "DOB": "1999-12-12"
8 }
```
- Response Status: `200 OK`, `188 ms`, `410 B`
- Response Body (Pretty JSON):

```
1
2   "ID": 5,
3   "CreatedAt": "2022-04-01T18:27:11.979-04:00",
4   "UpdatedAt": "2022-04-01T18:27:11.979-04:00",
5   "DeletedAt": null,
6   "userId": 0,
7   "firstname": "firtnname",
8   "lastname": "latnaoe",
9   "email": "fiaime.lannnaoe@gmail.com",
10  "profession": "Student",
11  "DOB": "1999-12-12T00:00:00Z"
```

2) Login:

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

localhost:8000/login

POST localhost:8000/login

Body (JSON)

```

1 {
2   "email": "fiaime.lastname@gmail.com",
3   "password": "Passwprid@001"
4 }

```

Headers (5):

- Content-Type: application/json
- Content-Length: 26
- Host: localhost:8000
- Connection: keep-alive
- Accept: */*

Response:

```

1 {
2   "message": "login success",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJFeHBpcmVzQXQiOjE2NDg5NDc10TEsIk1zc3VlZEF0IjoxNjQ40DYxMTkxLCJJc3N1ZXII0iJHYXRvck1hcnQiLCJTdWJqZWNO1joiVG9rZW4gZm9yIEhdhdG9yTWFydCBmcn9udGVuZCIsImF1dGhvcml6ZWQiOnRydWUsInVzZXJlWFpbCI6ImZpYW11Lmxhc3RuYW1lQGdtYWlsLmNvbSIsInVzZXJpZCI6MiwiDXNlc5hbWUiOjmaXJ0bmFtZSJ9.pP0G1Lp-q9YMkGmEnykbaiolN6wbwnDakXpRtLj0GNg"
4 }

```

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJFeHBpcmVzQXQiOjE2NDg5NDc10TEsIk1zc3VlZEF0IjoxNjQ40DYxMTkxLCJJc3N1ZXII0iJHYXRvck1hcnQiLCJTdWJqZWNO1joiVG9rZW4gZm9yIEhdhdG9yTWFydCBmcn9udGVuZCIsImF1dGhvcml6ZWQiOnRydWUsInVzZXJlWFpbCI6ImZpYW11Lmxhc3RuYW1lQGdtYWlsLmNvbSIsInVzZXJpZCI6MiwiDXNlc5hbWUiOjmaXJ0bmFtZSJ9.pP0G1Lp-q9YMkGmEnykbaiolN6wbwnDakXpRtLj0GNg
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE
{ "alg": "HS256", "typ": "JWT" }
PAYLOAD: DATA
{ "ExpiresAt": 1648947591, "IssuedAt": 1648861191, "Issuer": "GatorMart", "Subject": "Token for GatorMart frontend", "authorized": true, "useremail": "fiaime.lastname@gmail.com", "userid": 2, "username": "firstname" }

3) Productsave:

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

The screenshot shows the Postman application interface. At the top, it displays the URL `localhost:8000/product`. Below the header, there's a toolbar with `Save`, `Send`, and other options. The main area shows a `POST` request to `localhost:8000/product`. The `Body` tab is selected, showing the following JSON payload:

```
8 "city": "gainesville fl",
9 "state": "florida",
10 "location_lat": "maps",
11 "location_long": "maps",
12 "target": "student",
13 "category": "Mobile",
14
15 "posted_date": "3rdFeb",
```

Below the body, the `Test Results` section shows a successful response with status `200 OK`, time `12 ms`, and size `669 B`. The response body is displayed in pretty print:

```
14 "location_lat": "maps",
15 "location_long": "maps",
16 "target": "student",
17 "category": "Mobile",
18 "posted_by": 2,
19 "posted_date": "3rdFeb",
20 "condition": "Like new",
21 "age": 1,
22 "status": "available",
23 "images": "images"
24
```

4) Category Dropdown:

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 categories from the dropdown. It throws an error if the value does not present in the dropdown.

The screenshot shows two API requests in Postman:

localhost:8000/categories

Method: GET
URL: localhost:8000/categories
Headers: Authorization (green dot), Headers (7)
Query Params:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body: 200 OK 5 ms 237 B | Save Response

Pretty Raw Preview Visualize JSON

```
1 [ "Automobile", "Mobile", "ElectronicsAppliances", "Furniture", "Books", "Sports", "Pets" ]
```

localhost:8000/product

Method: POST
URL: localhost:8000/product
Headers: Authorization (green dot), Headers (9)
Body: 400 Bad Request 7 ms 181 B | Save Response

Params: none
Body: JSON
Body Content:

```
8 "city": "gainesville fl",
9 "state": "florida",
10 "location_lat": "maps",
11 "location_long": "maps",
12 "target": "student",
13 "category": "Sample",
14
15 "posted_date": "3rdFeb",
```

Body: Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 "Invalid Category"
```

5) Filter Products:

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

The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8000/filterproducts`. The method is selected as `POST`. The `Body` tab is active, showing the following JSON payload:

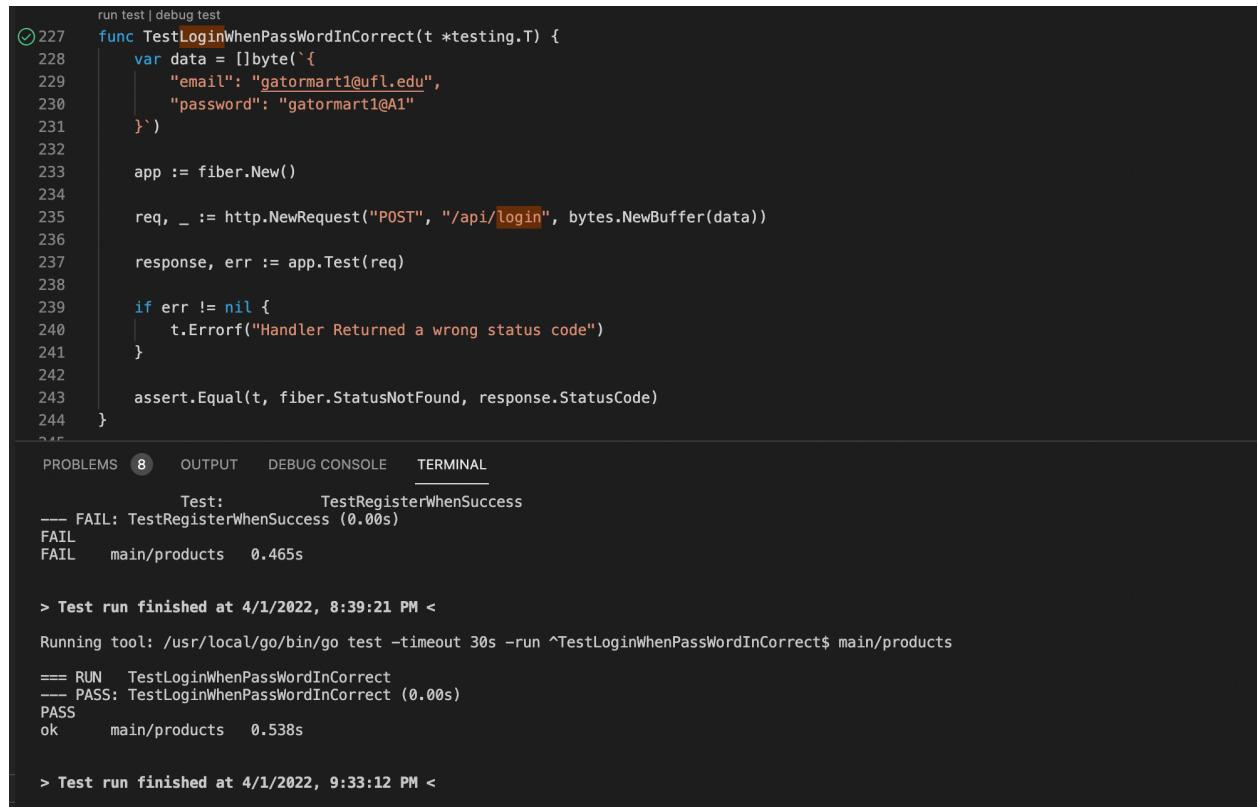
```
2   "title": "Fridge",
3   "condition": "",
4   "target": "",
5   "age": {
6     ...
7   },
8   "sortBy": "",
9   "price": {
10     ...
11     "operator": "between",
12     "fromvalue": 20,
13     "tovalue": 30
14 }
```

Below the body, the response status is shown as `200 OK` with `6 ms`, `1.82 KB` size, and a `Save Response` button. The response body is displayed in `Pretty` format:

```
1 [
2   {
3     "ID": 3,
4     "CreatedAt": "2022-03-31T09:54:19.666-04:00",
5     "UpdatedAt": "2022-03-31T12:32:27.585-04:00",
6     "DeletedAt": null,
7     "title": "Refrigerator Mini Beer Beverage Wine Soda Fridge Glass Door 115 Can Drink Cooler",
8     ...
9   }
10 ]
```

6) Unit test Cases:

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



The screenshot shows a code editor with a Go file containing test cases. The code defines a test function `TestLoginWhenPassWordInCorrect` that sends a POST request to the `/api/login` endpoint with incorrect credentials and asserts that the response status is `fiber.StatusNotFound`. Below the code is a terminal window showing the execution of the tests. The output indicates a failure for the `TestRegisterWhenSuccess` test and a success for the `TestLoginWhenPassWordInCorrect` test. The total execution time is 0.465s.

```
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 <

Backend Video:

https://drive.google.com/file/d/1jW3R8ptlj_4cBiZor8pP9m3OgjSceNq/view?usp=sharing

SWAGGER API Doc:

<https://github.com/bhanupalagati/GatorMart/blob/main/Swagger%20API%20DOCUMENTATION.pdf>