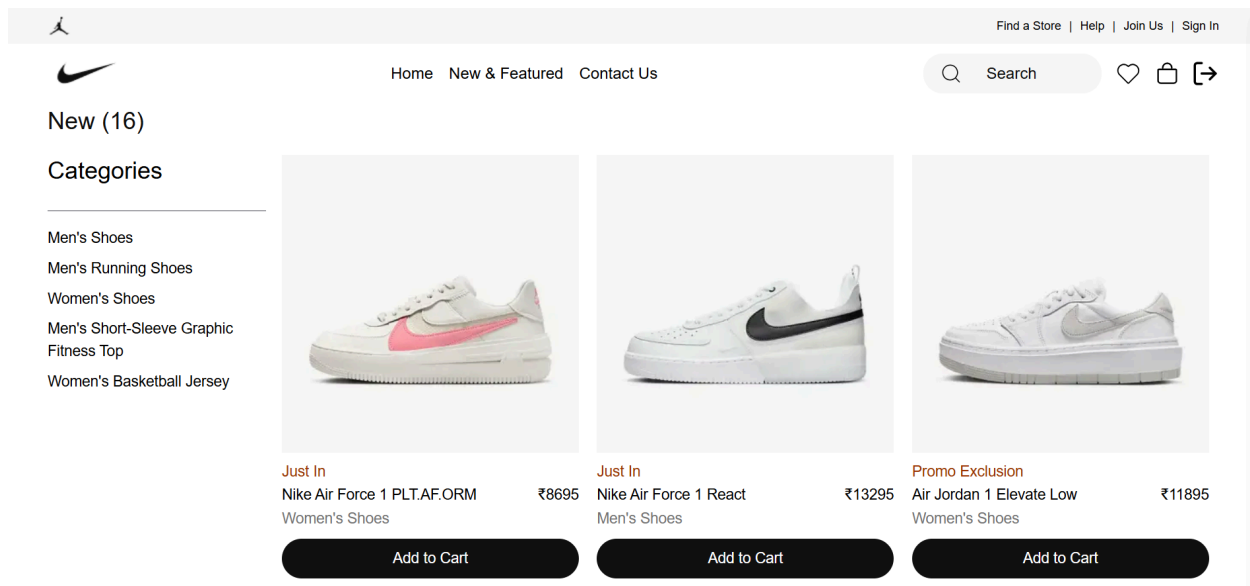


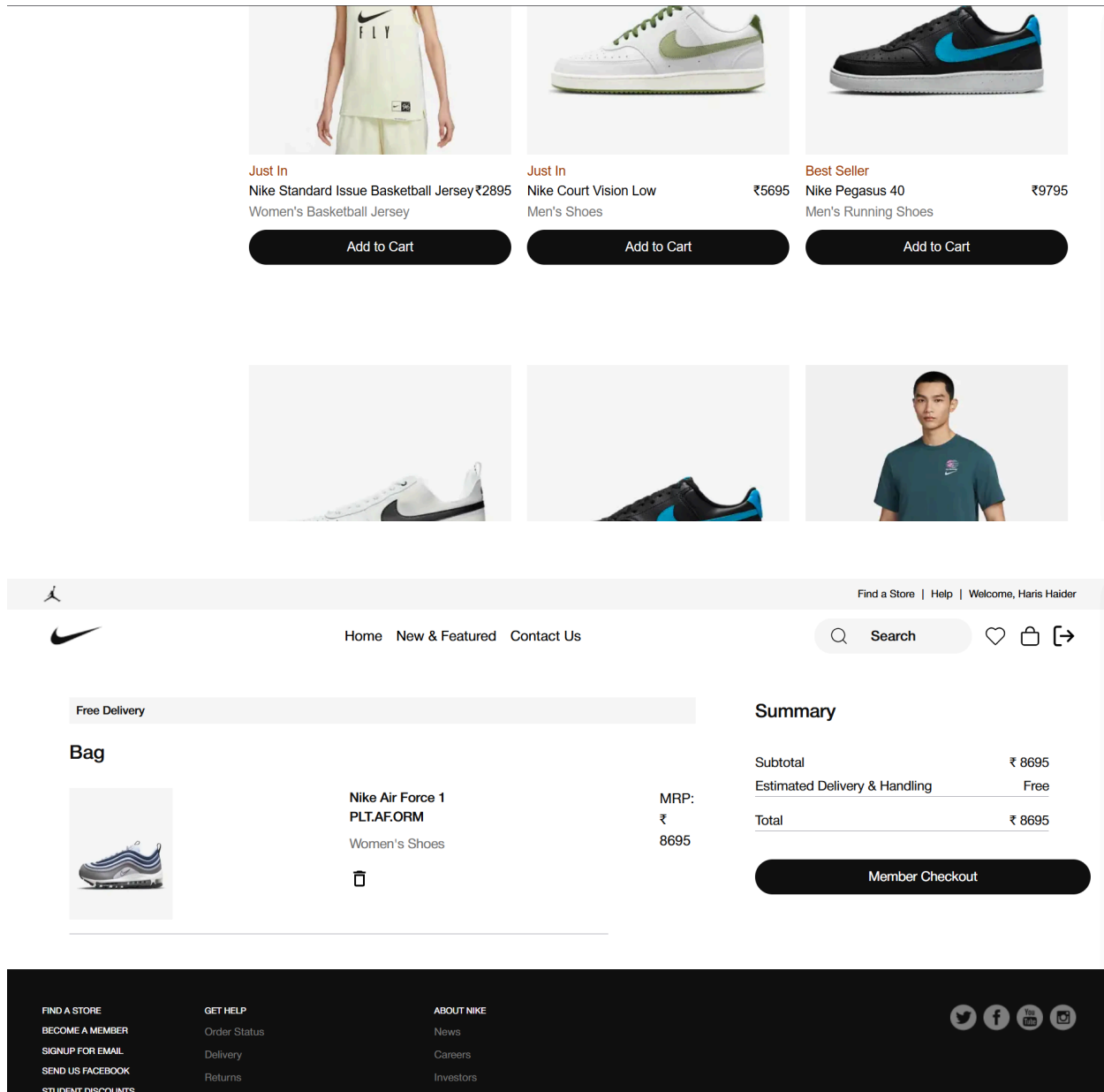
Testing, Error Handling and Backend Integration

The task for day 5 was to focus on testing, error handling with backend integration following are the steps i did during day 5:

1. Functional Testing:

- Task: To Check that all the features of a marketplace is properly working or not.
 - ☐ Checked All the products are listed correctly with dynamic data like name, price, image, tags etc.
 - ☐ Check Product Details pages are correctly set up and showing dynamic data coming from Sanity CMS.
 - ☐ Tested addToCart functionality is properly working with subTotal price and products.





2. Error Handling:

- Implemented Error Handling at products fetching, details and in cart adding functionalities.
- Result: React-Hot-Toast will notify with both error and success message.

This screenshot shows the Visual Studio Code editor with a project named 'FIGMADESIGN'. The Explorer sidebar on the left shows a file tree with folders like 'actions', 'api', 'auth', 'delete', 'order', 'products', 'signup', 'user', 'cart', 'checkout', 'contact', 'login', 'products', and 'register'. The 'products' folder is expanded, and 'route.ts' is selected. The main editor displays the code for a GET route handler in 'route.ts'.

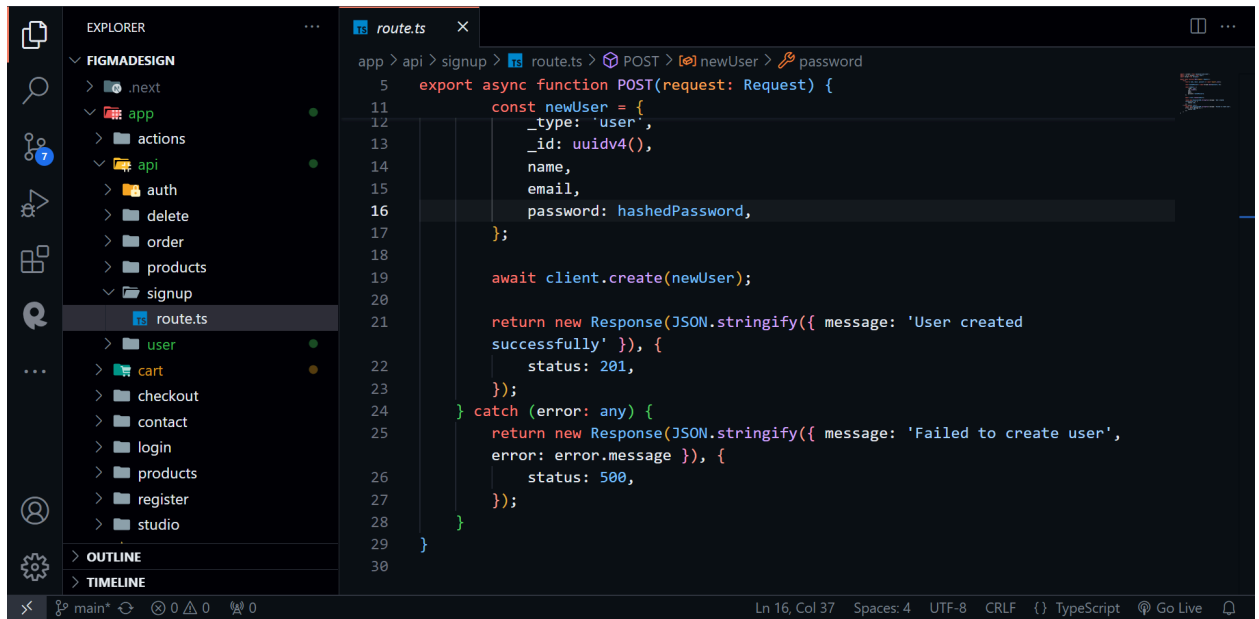
```
app > api > products > route.ts > GET
4  export async function GET(request: Request) {
6      const query = `
16         image: {
17             description
18         }
19     `;
20
21     const data = await client.fetch(query);
22
23     return NextResponse.json(data);
24 } catch (error) {
25     return NextResponse.json(
26         { error: "Failed to fetch products. Please try again later." },
27         { status: 500 }
28     );
29 }
30 }
```

The status bar at the bottom indicates 'Ln 29, Col 6', 'Spaces: 4', 'UTF-8', 'CRLF', and 'TypeScript'.

This screenshot shows the Visual Studio Code editor with the same 'FIGMADESIGN' project. The Explorer sidebar shows the 'order' folder expanded, and 'route.ts' is selected. The main editor displays the code for a POST route handler in 'route.ts'.

```
app > api > order > route.ts > POST > addedProduct > productName
3  export async function POST(request: Request) {
13      const addedProduct = await client.create({
19          image: {
20              _type: 'image',
21              asset: {
22                  _type: 'reference',
23                  _ref: image,
24              },
25          },
26      });
27
28      return new Response(JSON.stringify({ message: 'Product added to cart
29      successfully', product: addedProduct }));
30  } catch (error: any) {
31      return new Response(JSON.stringify({ message: 'Failed to add product to
32      cart', error: error.message }));
33  }
34 }
```

The status bar at the bottom indicates 'Ln 16, Col 20', 'Spaces: 4', 'UTF-8', 'CRLF', and 'TypeScript'.

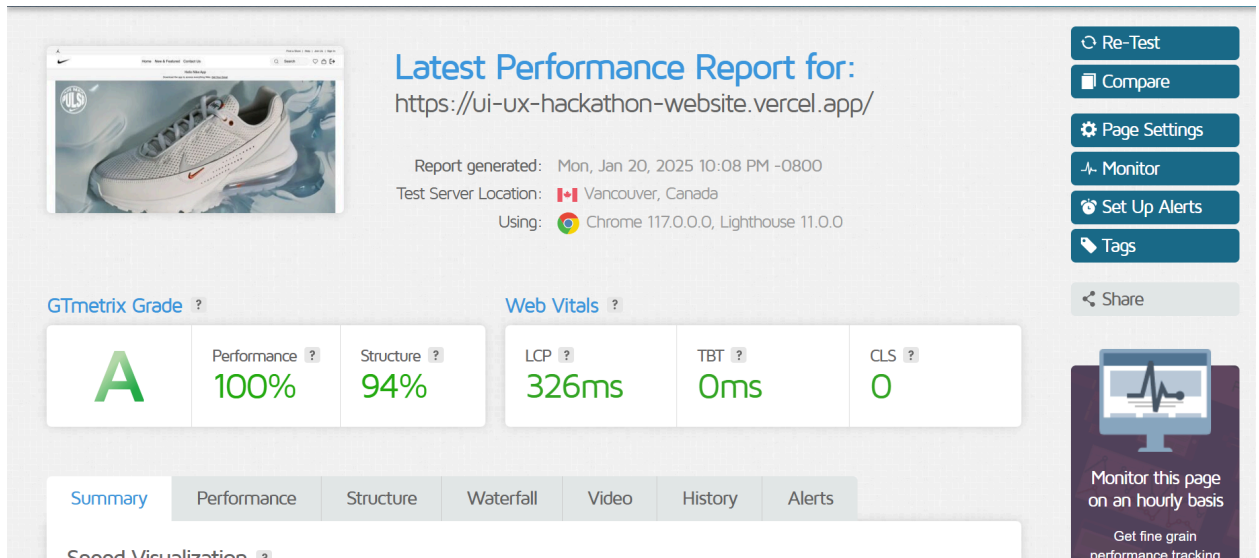


The screenshot shows the Visual Studio Code editor with a project named 'FIGMADESIGN'. The Explorer sidebar on the left shows a file tree with folders like 'actions', 'api', 'auth', 'delete', 'order', 'products', and 'signup'. Inside 'signup', there is a file 'route.ts' which is currently selected and open in the main editor. The code in 'route.ts' is a TypeScript file defining a POST endpoint for creating a new user. It uses the 'express' and 'mongoose' libraries. The code defines a 'newUser' object with fields '_type', '_id', 'name', 'email', and 'password'. It then calls 'client.create(newUser)' to create the user. If successful, it returns a 201 status with a success message. If an error occurs, it returns a 500 status with an error message. The status bar at the bottom indicates the file is at line 16, column 37, using UTF-8 encoding and CRLF line endings.

```
5 export async function POST(request: Request) {
11   const newUser = {
12     _type: 'user',
13     _id: uuidv4(),
14     name,
15     email,
16     password: hashedPassword,
17   };
18
19   await client.create(newUser);
20
21   return new Response(JSON.stringify({ message: 'User created
22     successfully' })), {
23     status: 201,
24   });
25 } catch (error: any) {
26   return new Response(JSON.stringify({ message: 'Failed to create user',
27     error: error.message })), {
28     status: 500,
29   });
30 }
```

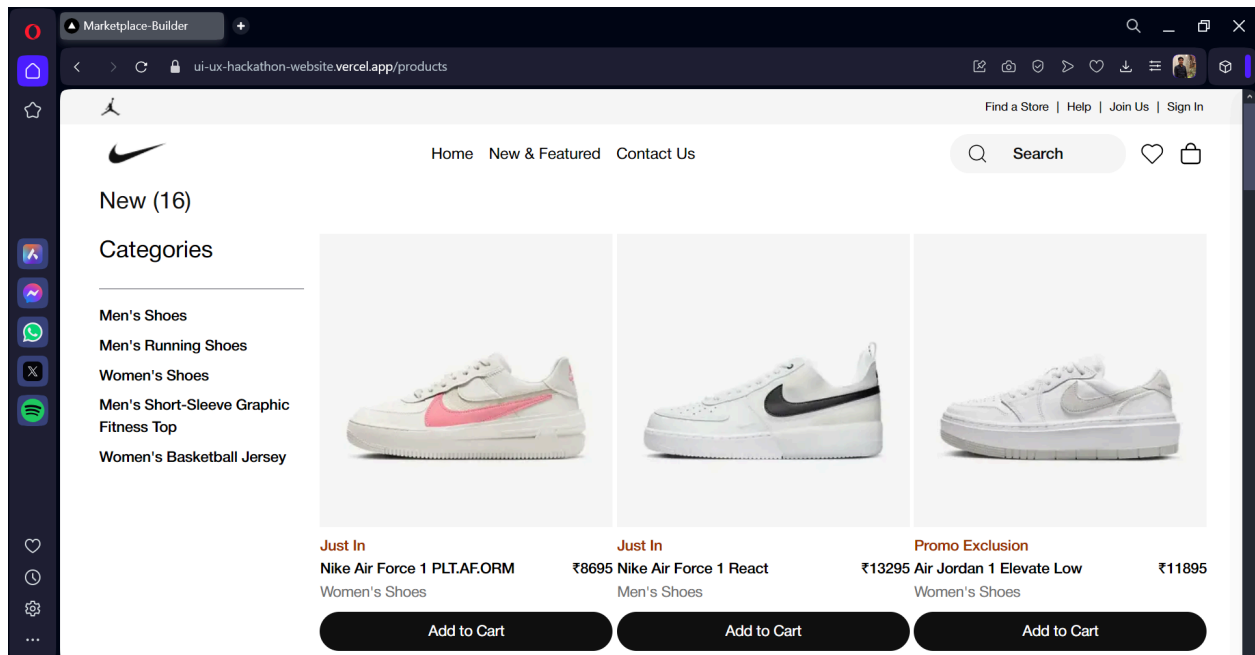
3. Performance Testing:

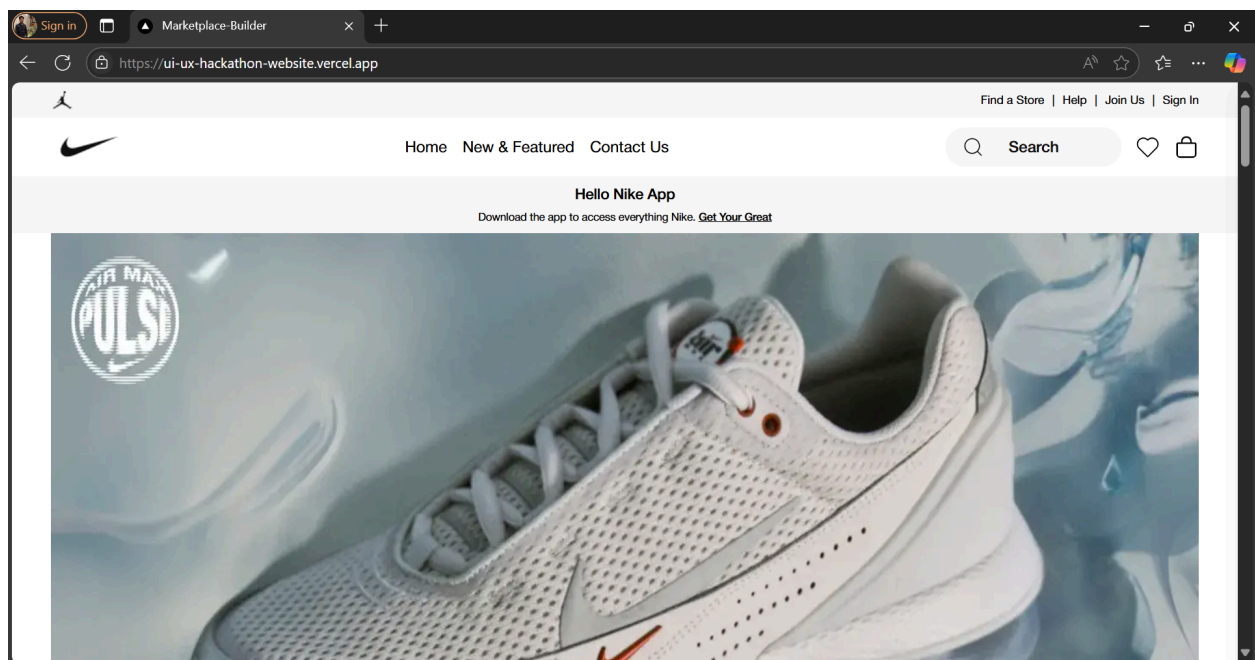
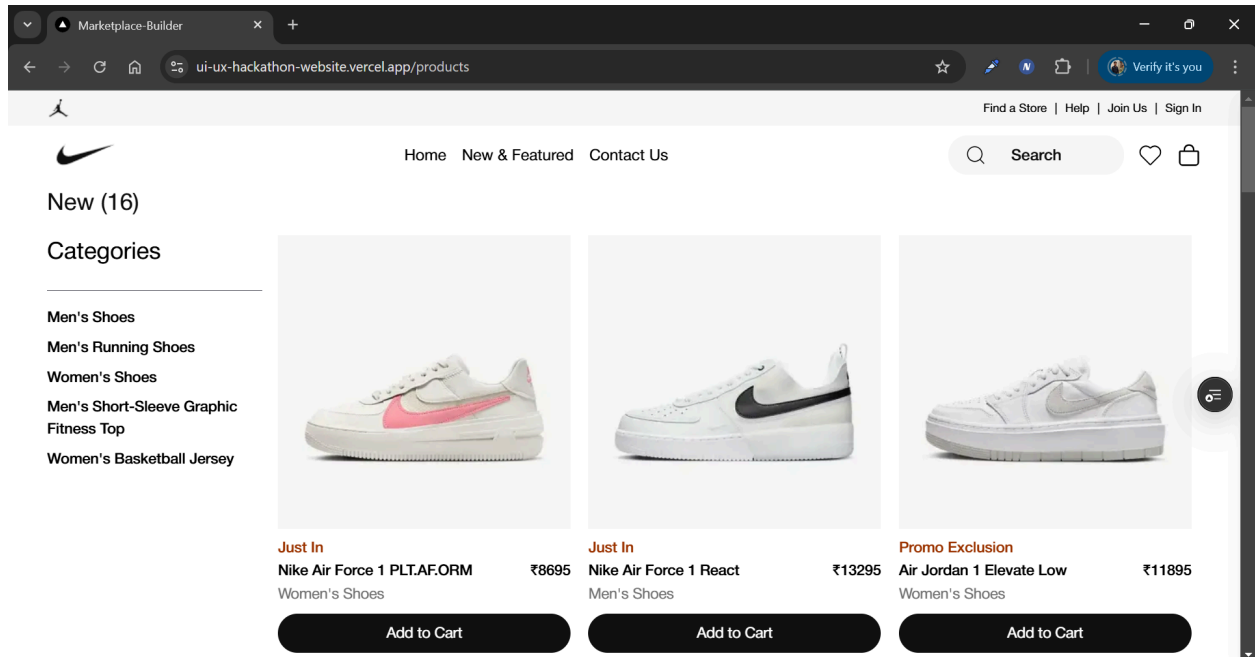
- I Tested performance of a website on a platform called gtmetrix.com



4. Cross Browser and Device Testing:


- Task: Ensure capability and speed across different browsers.
 - ☐ Tested on Chrome, Brave, Opera and Edge browser.
 - ☐ Tested on mobile phones two different resolution laptops and one tablet.





5. Security Testing:

- Marketplace is secured by authentication using Auth.js only authenticated users with all details can place an order.
- Also Implemented middleware to enhance security.




Find a Store | Help | Join Us | Sign In

HomeNew & FeaturedContact Us

QSearch

♡🛒



**YOUR ACCOUNT
FOR EVERYTHING
NIKE**

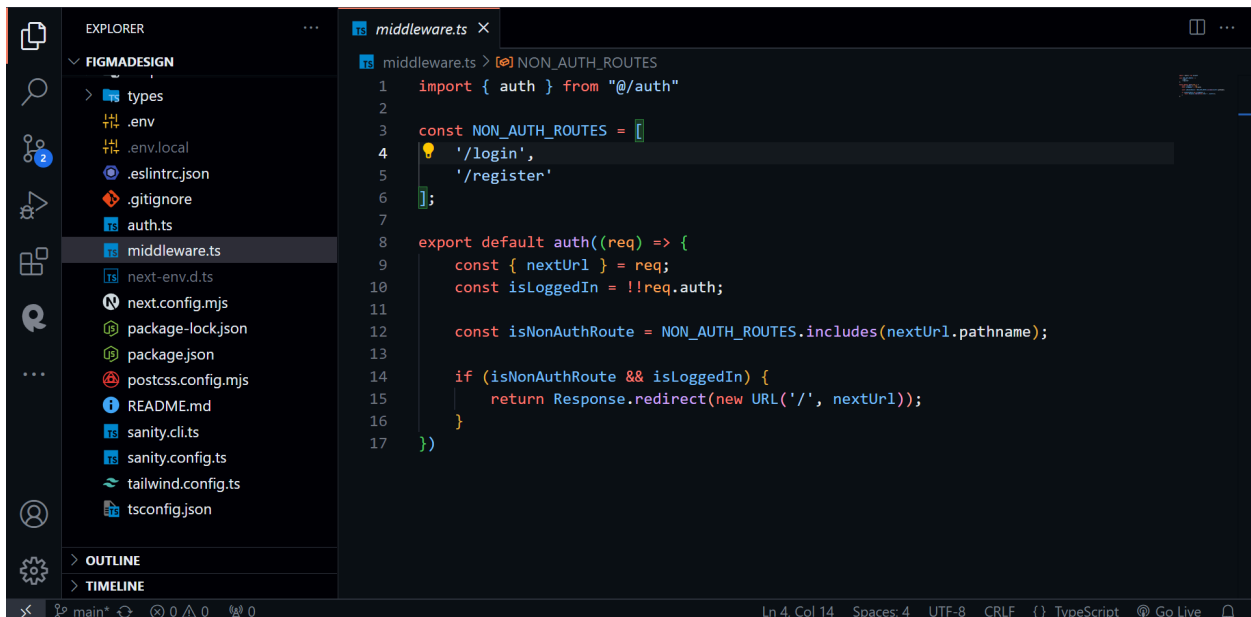
Email Address

Password

By logging in, you agree to Nike's [Privacy Policy](#) and [Terms of Use](#).

SIGN IN

Not a Member? [Join Us](#).



6. User Acceptance Testing (UAT):

- Collected feedback from non tech users.

Feedback:

- ☐ Got positive feedback with few enhancements.

7. Documentation Updates:

- Updated documentation to reflect changes in schema and functionalities.

Updated Schema according to use case:

```
export const productSchema = {
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'productName',
      title: 'Product Name',
      type: 'string',
    },
    {
      name: 'category',
      title: 'Category',
      type: 'string',
    },
    {
      name: 'price',
      title: 'Price',
      type: 'number',
    },
    {
      name: 'inventory',
      title: 'Inventory',
      type: 'number',
    },
    {
      name: 'colors',
      title: 'Colors',
      type: 'array',
      of: [{ type: 'string' }],
    },
    {
      name: 'status',
      title: 'Status',
      type: 'string',
    },
    {
      name: 'image',
      title: 'Image',
    }
  ]
}
```



```
        type: 'image',
        options: {
            hotspot: true,
        },
    },
    {
        name: 'description',
        title: 'Description',
        type: 'text',
    },
],
}
```

Conclusion:

- Today's task updated my application and enhance its features and functionality.