مجلة بحوث جامعة تعز
Taiz University Research Journal
العدد(38)
Issue(38)
TURJ
ISSN : 7848-2958
pp. 116 - 135
ديسمبر 2023م
December 2023

ELECTRONIC COMMERCE PLATFORM AS SINGLE PAGE APPLICATION USING REACT JS AND LARAVEL FRAMEWORK

Mogeeb A. Saeed[1]
[1]Associate Professor at Computer Networks And Cybersecuirty Department Al-saeed Faculty of Engineering & IT
Taiz University, Yemen
mogeeb1982@gmail.com

Amr A. Saeed[2]
[2]Engineer at Information Technology Department Al-saeed Faculty of Engineering & IT
Taiz University, Yemen
amranes047@gmail.com

Shehab M. Alward[3]
[3]Engineer at Information Technology Department Al-saeed Faculty of Engineering & IT
Taiz University, Yemen
shehab.m.alward@gmail.com

Abstract – The current era has witnessed many developments in many fields of networks and internet in many aspects. The most important of which is the business aspects and web development where many developing frameworks appears that increase the efficiency, speed, usability, and security of web applications. In this paper, we aim to Enhancing performance, efficiency and reliability of e-commerce platform and Develop a responsive design that is compatible with all types of devices, user-friendly that is visually appealing and easy to navigate and give the best user experience. For this reason we used modern libraries Such as the React library, Redux and tailwindcss framework for building user interfaces and the Laravel framework for backends. And using MySQL as a database. The platform consists of three parts. The first part is the administration part which allows admins to have control over this platform and make them able to get statistics of the shop, control the products, orders, customers, and more. Second, the employee part. which allows different markets to add their products to the shop and show their statistics and orders that come to them. Finally, the customer part. which provide interfaces for displaying different products and providing the browser of these products with a good amount of information that helps him choose the appropriate product more accurately.

Index Terms: Single Page Applications, ReactJS, Redux, Laravel, REST API, E-commerce, frontend-backend.

I. Introduction

The current era has witnessed many developments in many fields of networks and internet in many aspects. The most important of which is the business aspects. Where the process of shopping and commercial transactions is done remotely via the internet easily. This has led to the need for Enhancing speed and reliability of e-commerce platform and Develop a user-friendly design that is visually appealing and easy to navigate. For this reason we used the best and latest technologies and tools to enhance both reliability and performance of e-commerce platform and Develop a user-friendly design that is visually appealing and easy to navigate and give the best user experience.

To develop a web application there are mainly two ways for develop web applications Single Page Applications (SPA), and Multi-Page Applications (MPA). "A single page application (SPA) is a web development approach that functions within a single web page. Unlike traditional multi-page applications, SPAs dynamically update the content on the same page without requiring

separate page loads for different interactions. This approach enhances the user experience by providing a seamless and responsive interface. [2].

Single Page Applications (SPAs) offer several advantages that make them a popular choice for web development [2]:

1. Enhanced Performance: SPAs deliver faster performance by loading all the necessary resources initially. Subsequently, when interacting with the page, only the required data is modified, minimizing unnecessary data transfers and improving overall responsiveness.
2. Caching and Offline Support: SPA can utilize browser caching mechanisms to store static assets within the client's browser. This feature enables the application to function offline or with limited connectivity by serving cached content and data from the client's local storage.
3. Easy Debugging: SPAs are built and developed using popular frameworks like React, Vue.js, and AngularJS, which provides excellent debugging and testing capabilities. [2].

In this project we used a single page application framework called ReactJS to build the front end of our platform and in the back end we used a PHP framework called Laravel.

II. Literature Review

A. JavaScript Frontend Frameworks

the literatures, the structure, extensive assessment, detailed analysis of the most popular JavaScript frameworks were explored in meticulous study [1]. comprehensive study of the types of web applications SPA and MPA, differences between them and the analysis of frontend frameworks for both types of applications was done [2]. The advantages and disadvantages for each framework under commercial criteria and potential growth of front end development in e-commerce was presented [3]. Detailed study of the web services which is implemented through SOAP and REST, the differences in the structure and performance comparison between the two was gathered. [4]

B. RESTFul API

RESTful web service is a popular architectural style for designing and implementing web APIs that allow different software applications to communicate and interact with each other over the internet. RESTful web services utilize standard HTTP Request methods like GET, PUT, POST, DELETE to communicate and perform operations on data [5]. RESTful web services use standards of the World Wide Web HTTP and XML, JSON for data representation [6], they are compatible with all types of interfaces or platforms it was phone, desktop, web. [7].

C. Single-Page Application

The Single Page Application (SPA) is a web application architecture that load information only on one page. aims to provide a smooth and interactive user experience by dynamically updating the current page that is, updating the part of the page where the change occurred instead of loading new pages entirely from the server. [8]. In this case, the server side updates certain parts based to user requests, and therefore SPAs can deliver high performance compared to traditional multi-page applications. [9,10].

III. Related Works

1) S. Tenzin, T. Lhamo, [11]."Design and Development of E-Commerce Web Application" This is an e-commerce web application that enables customers to shop online without the need to visit the physical store. Its primary objective is to streamline the shopping process, reduce salesperson workload, and eliminate manual errors by automating record entry. By utilizing this system, customers can significantly reduce costs and save time that would otherwise be wasted. This enhanced level of service not only attracts new customers but also fosters customer loyalty, contributing to long-term business growth.

2) Nham, Tran[12]."Developing an E-commerce application prototype with React JS and Firebase" This thesis focuses on the development process of an e-commerce application prototype utilizing ReactJS and Firebase services. It aims to provide a comprehensive understanding of e-commerce, while also exploring the fundamental aspects of ReactJS and Firebase. The central objective is to establish a conceptual framework leveraging Firebase tools and services. The ultimate outcome of this project, based on the thesis, is a fully functional prototype of an e-commerce webstore.

3) DAnh. Le [13]. "E-Commercial Full Stack Web Application Development" This thesis aims to develop a web application that seamlessly integrates front-end and back-end technologies to optimize e-commerce processes. By leveraging a combination of software frameworks and programming languages, the application establishes a robust and feature-rich platform. This research focus on the challenges associated with designing full-stack e-commerce applications and proposes effective strategies to overcome these obstacles.

4) Kankaala. M [14]. "Enhancing E-Commerce with Modern Web Technologies". This thesis aimed to explore the latest advancements in web development and apply them to the

creation of an innovative e-commerce web application. The implemented application adopts a headless architecture, effectively separating the backend and frontend logic. This architectural approach enables the optimization of the frontend components and facilitates the development of a progressive web application.

5) N. Mai [15]. "E-commerce Application using MERN stack" This thesis focuses on studying the fundamental components of the "MERN Stack technology, including MongoDB, ExpressJS, ReactJS, NodeJS", and their application in building an e-commerce web application. The thesis discusses essential functionalities such as user registration, login, dashboard presentation, store categories, and product display. Furthermore, it explores the integration of Braintree as a payment gateway, allowing users to search for product stores. Additionally, the thesis covers the development of administrative functions such as user management, store management, and generating statistics and reports.

IV. Proposed Method

In this paper focus to build e-commerce application as single page application that work on web by use ReactJS to build the front end and can also compatible with all types of interfaces it was phone, desktop by Restful API that built using the Laravel framework.

The application comprises two primary components: the front end and the back end; the front end consists of react components and redux. The react components is the UI that displayed, by react components its possible for single page applications to display required content without requiring the browser to reload the entire page, the redux as central store to store all the states of the applications that are easily accessible by all the app components, the backend is the part that processes the requests, consists of Laravel and API. Laravel use to handle the routes and API which is the interface that associated the frontend with backend and the idea of use API is

building an application compatible with all types of interfaces it was phone, desktop, web. Figure 1 illustrates architecture for our model.

Architecture for proposed model consist from

1) Front end “react Js”
2) Redux store
3) Restful API
4) Back end

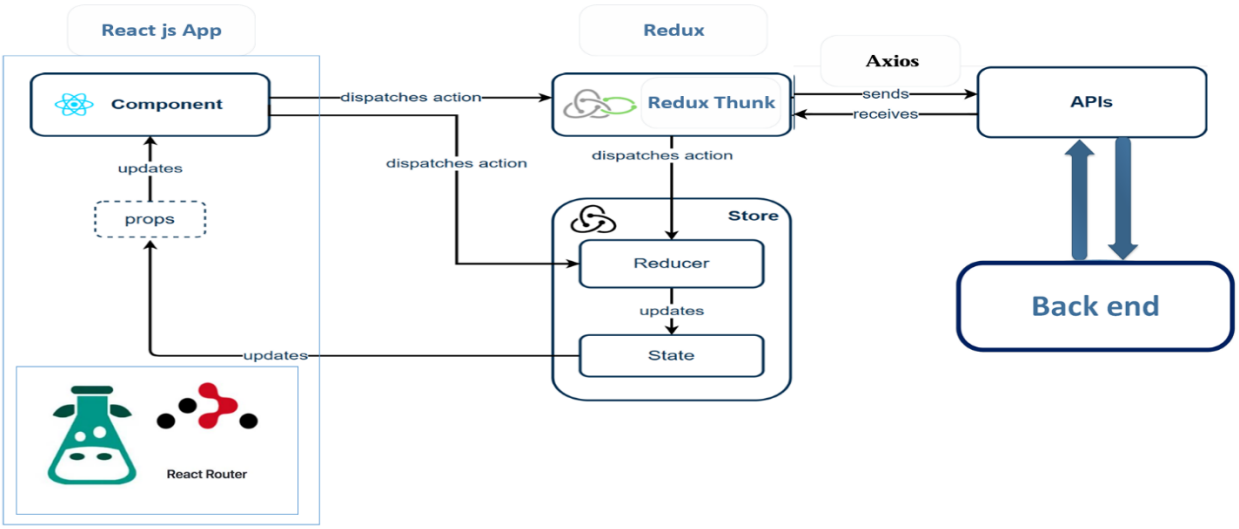

FIGURE 1. PROJECT ARCHITECTURE

A. Front end “React Js”

ReactJS is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is developed by Meta. With React, you can build web applications rapidly and efficiently, requiring considerably less code compared to traditional JavaScript development. React allows you to develop applications by creating reusable components, which can be viewed as independent building blocks. These components serve as individual elements of the overall user interface, and when combined, they form the complete interface of the application. [16]. one of the most important reasons for using React is compatibility and seamless integration with Redux which enables efficient state management, predictable data flow, and easier development of application.

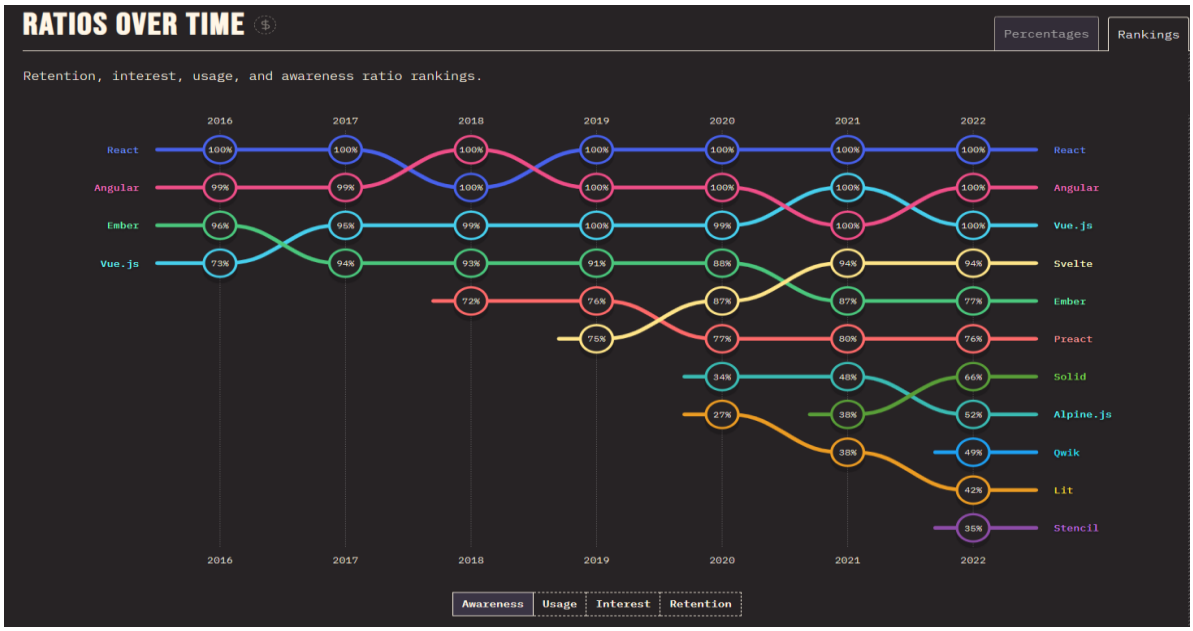

Figure 2. rating for frontend JS frameworks [17].

B. Redux store

"Is an open source state management JavaScript library for managing the application data. Allows all components of react JS to read the data from redux store and dispatch actions to the store to update the state. With Redux, we can store the state of entire application in a single store, making it easier to manage and access data across components. also Redux ensures a well-defined data flow, ensuring that modifications to the state can only occur when an action is created and dispatched through Redux. Therefore it makes it easy to understand how application data will be altered in response to user actions." [18]. we used redux as centralized store to store all the states of the applications that are easily accessible by all the app components. Each component does not have to pass the props around within the component tree to access those states.

C. RESTFul API

API stands for "Application Programming Interface," which is the interface to associated the frontend with backend. The idea of using the API to create an application that is compatible with all kinds of interfaces. Figures 4 shows screens of product API.

D. Back end

Laravel Architecture

"Laravel is a free and open-source PHP web framework, created by Taylor Otwell" [19], and is designed for building web applications that adhere to the model-view-controller (MVC) architectural pattern. Its primary goal is to simplify development by providing convenient features for common tasks encountered in web projects, including authentication, routing, sessions, and caching. We use Laravel for handle the routes, processing the requests.

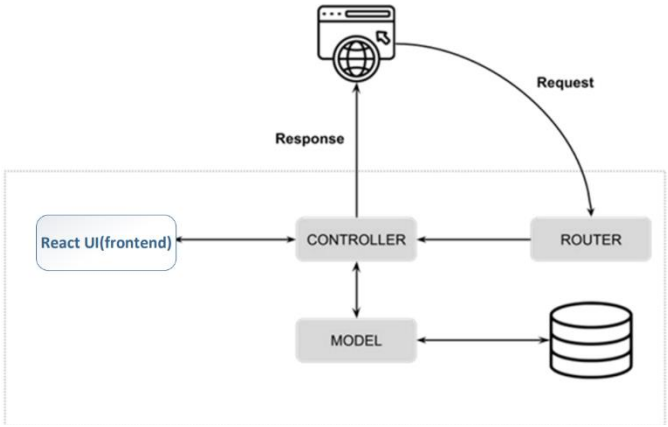

FIGURE 3. LARAVEL ARCHITECTURE

In Figures 3 show Laravel architecture. When the browser (client) make requests. the requests transmitted to the route (api.php), because the frontend is in react JS. The router is the most important components in MVC that initially receives the request from the client and allocates which Controller is going to handle the request. The router checks if the path or routes exists and is it associated with the controller. Then send the request to the controller. The Controller component work as mediator between the frontend and Model components. Its receive the requests or routes from the frontend, and collaborates with the model to query data or update existing data accordingly, and return the data as JSON to the frontend.

V. RESULTS

The User Interface Design

The application consists of two parts. The first part is the administration part which allows admins to have control over this platform and the customer part. which provide interfaces for displaying different products and providing the browser of these products with a good amount of information that helps him choose the appropriate product more accurately.

1. Administration Part DASHBOARD

The dashboard has been created to manage the entire site, like orders, employees, customers, products etc. This page presents the data in the form of charts, graphs, and tables that are easy to read and understand. This page displays the daily, weekly, monthly and yearly statistics for orders, sales, earnings and visitors that which enables the manager or the customer to formulate a future vision. Figures 5–8 show some Screenshot of the Dashboard.

Our application is characterized by high performance and efficiency, as it is possible to display, manage and control products, orders, customers, and employees with all flexibility. such as in Figure 10–11 user can group the orders based on specific column by drag and drop that column in the grid header. The user can also search, filter and order any column in the grid.

In addition to the profile page in figure 9.

Also in figure 12 shows some of add section which is add products.

2. Customer part

in Figure 13–16 we display fragments of user interface like login screen, home screen that contain products and categories and last offer, the product details screen that will appear when the user clicking on a product. and also The shopping cart that contain of all product that select by customer and can Make payment within it.


```
[
    {
        "id": 3,
        "product_type": "watches",
        "company": "apple",
        "model": "watch",
        "price": 1200,
        "supplier": "apple",
        "video": "0WmkbvyBG9umBpSwtbS74UARidPTWnsT8N4ZCVAC.mp4",
        "discount": 8,
        "total_amount": 10,
        "image_name": "4LyWuo85cdfLq8Jnn9XDPrlOedcOMAgoiqcCMOvy.
    },
    {
        "id": 2,
        "product_type": "monitors",
        "company": "apple",
        "model": "monitor",
        "price": 12000,
        "supplier": "apple",
        "video": "KyWwjNg3rFCQLdKOmugqsk5W3c3Kla1oTlb3NESJ.mp4",
        "discount": 15,
        "total_amount": 1,
        "image_name": "owKoPrN0mkwPq0p1pTg5DlPwuPhIegdWaD2KodQ5.
    },
```


Figure 4. Screenshot of product API "JSON'' format.

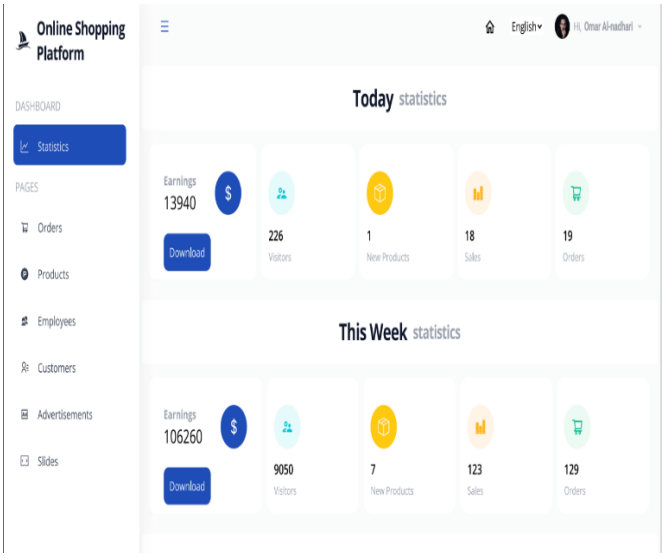

Figure 5. Screenshot of the daily and weekly statistics.

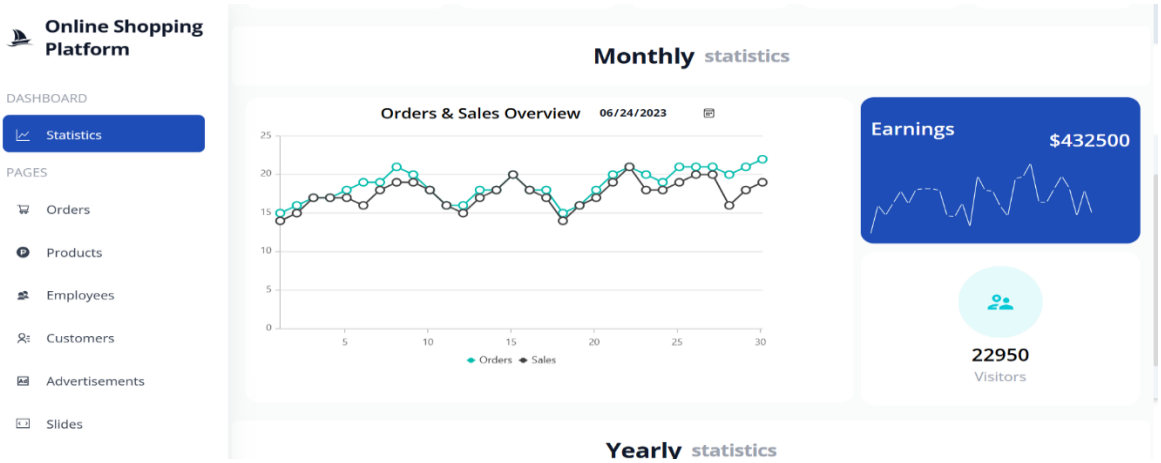

Figure 6. Screenshot of the monthly statistics.

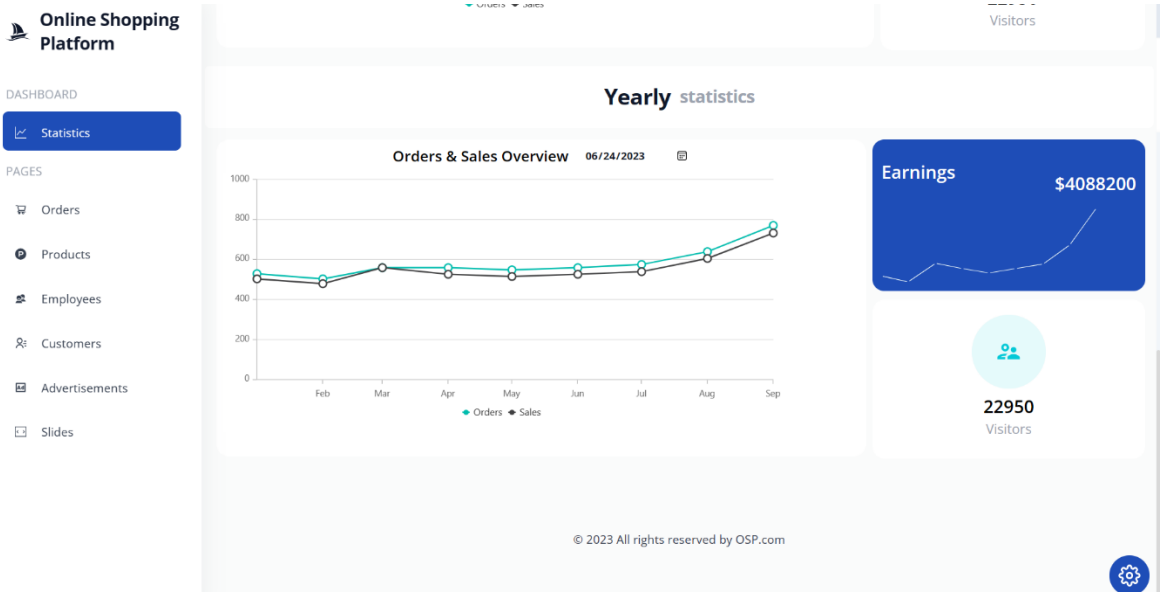

Figure 7. Screenshot of the yearly statistics.

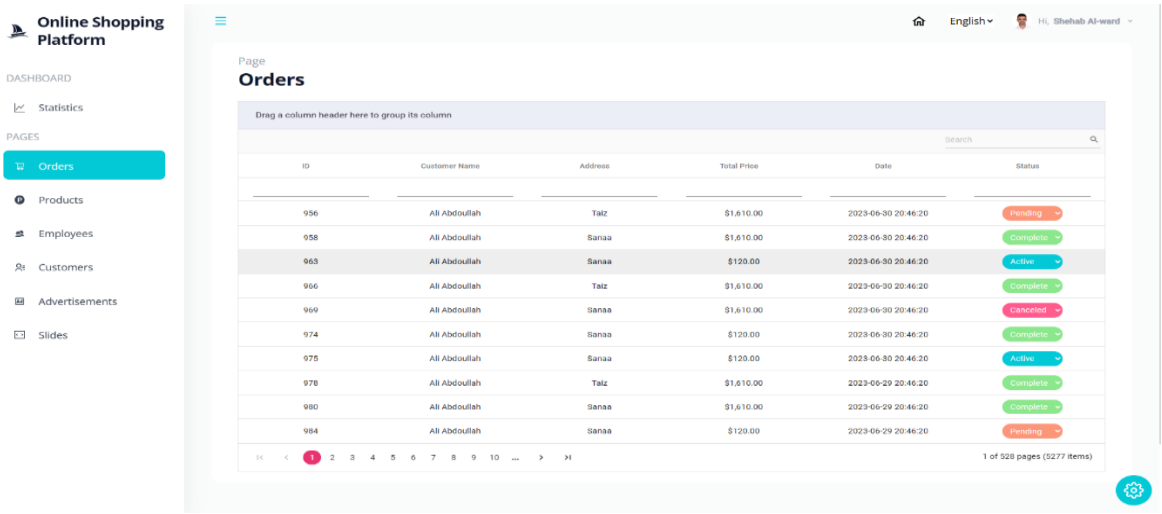

Figure 8. Screenshot of the orders page.

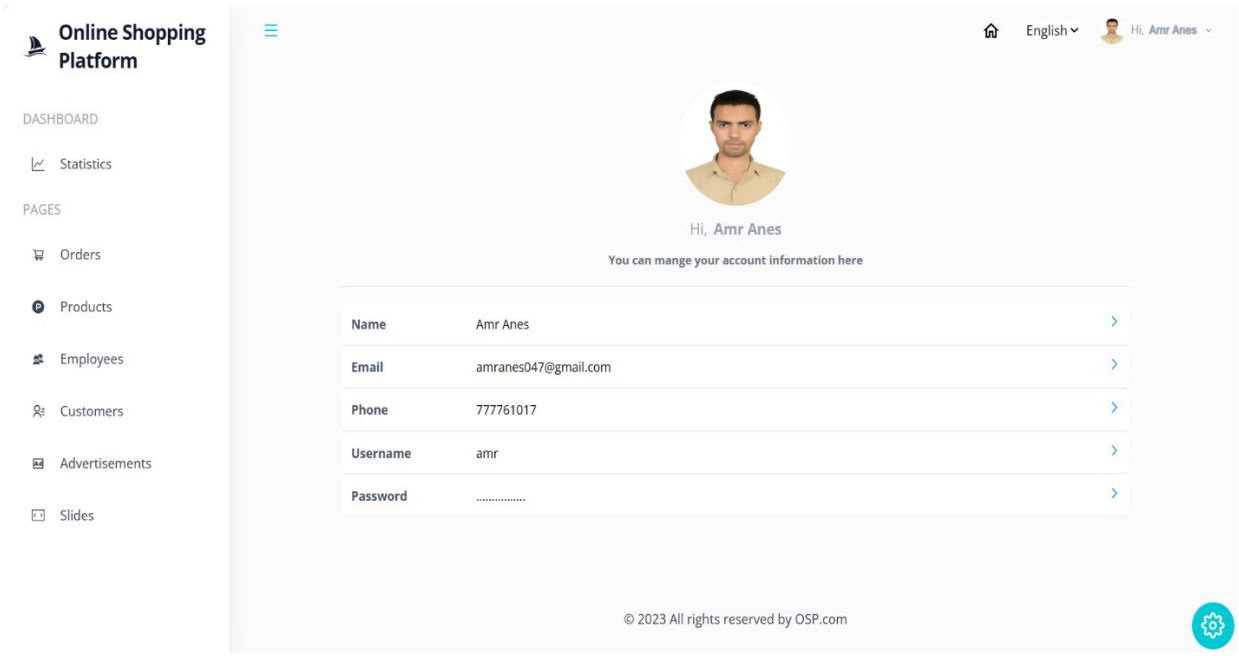

Figure 9. Screenshot of the profile page.

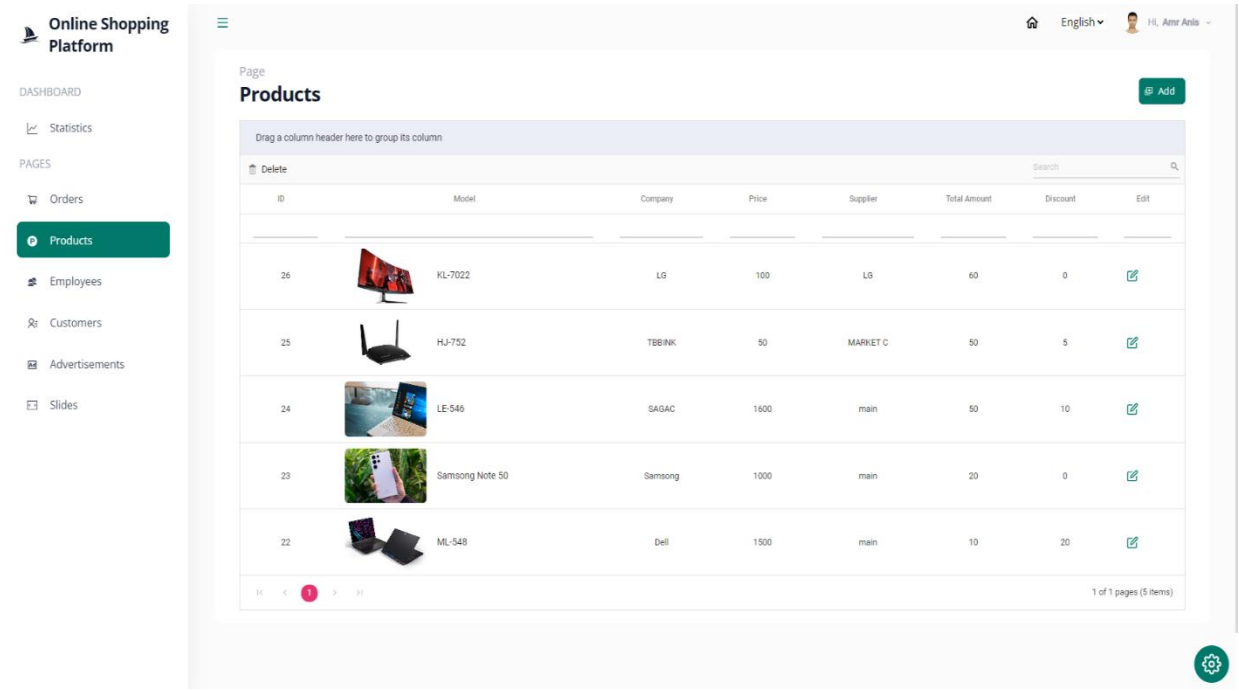

Figure 10. Screenshot of the products page.

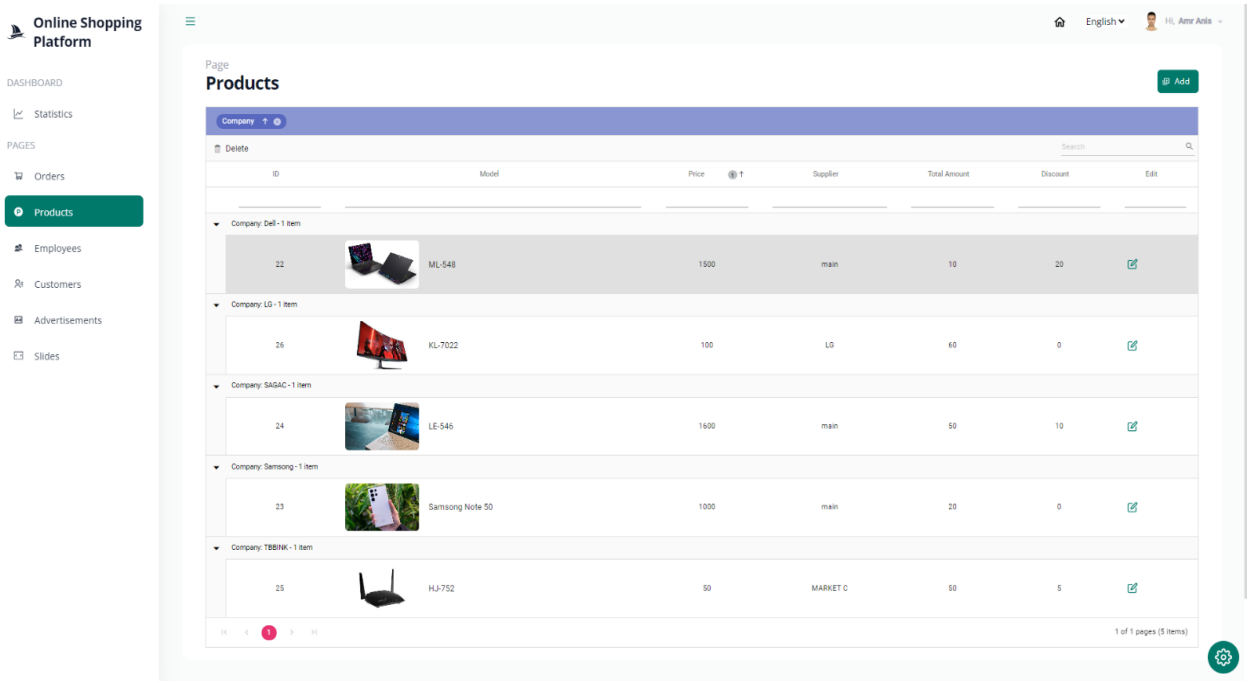

Figure 11. Screenshot of the grouping the products.

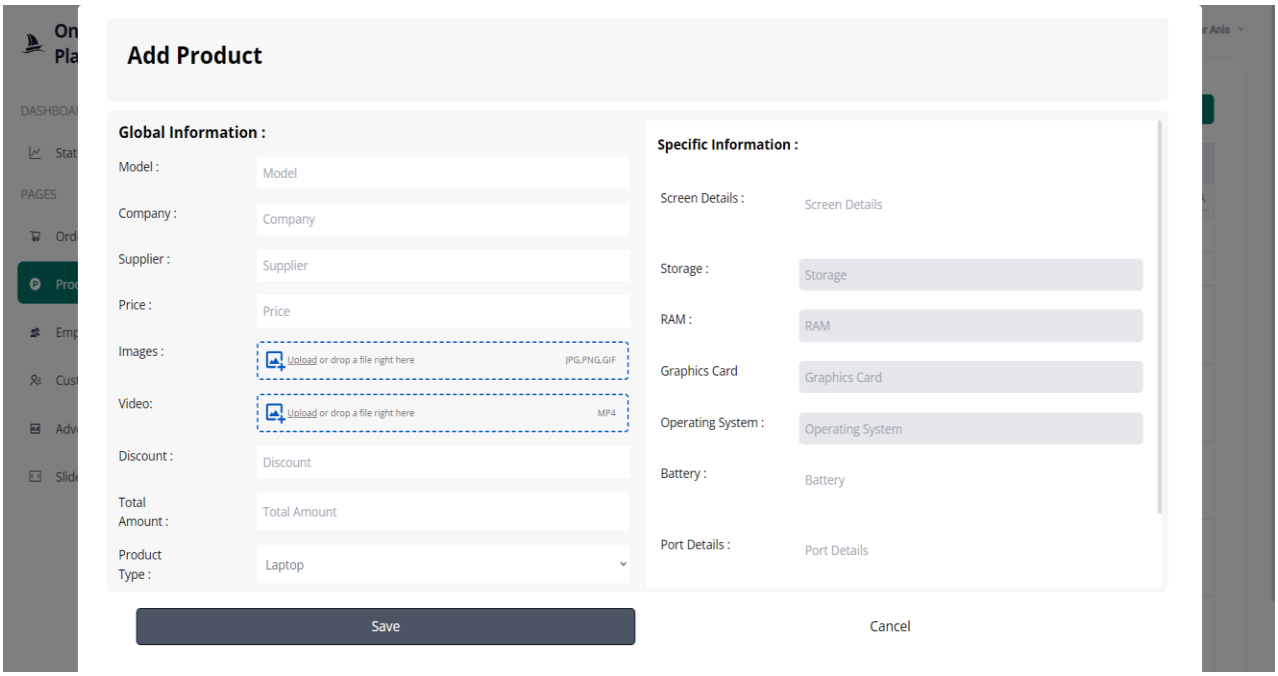

Figure 12. Screenshot of the add products page.

ديسمبر 2023م
December 2023
ISSN : 7848-2958
pp. 116 - 135
TURJ
العدد (38)
Issue (38)
مجلة بحوث جامعة تعز
Taiz University Research Journal

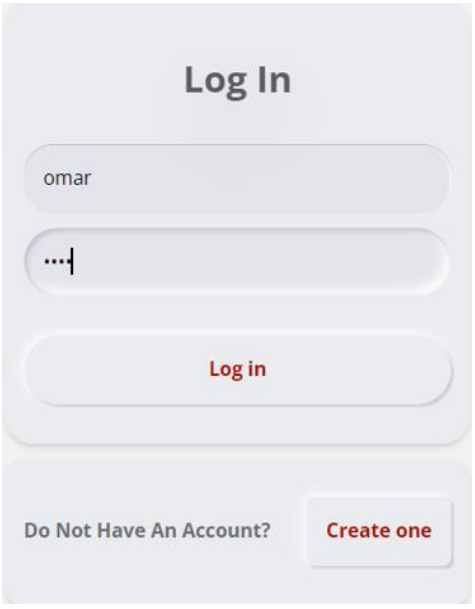

Figure 13. Screenshot of the login.

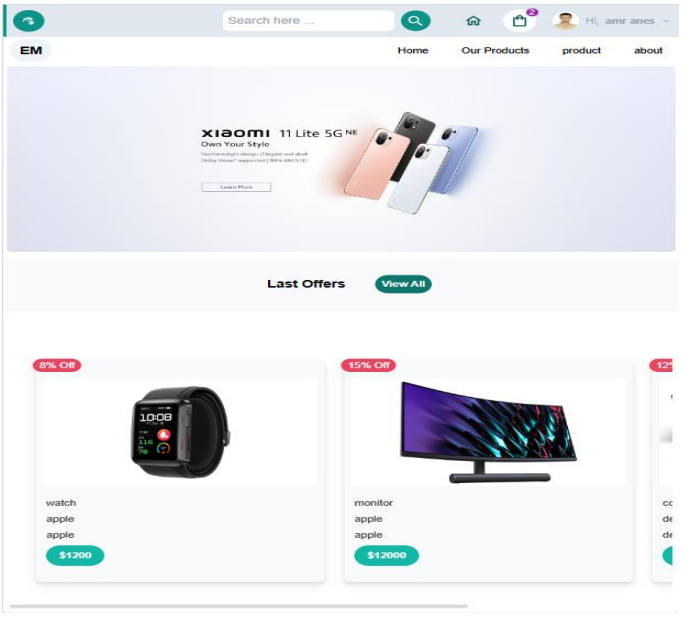

Figure 14. Screenshot of the home page.

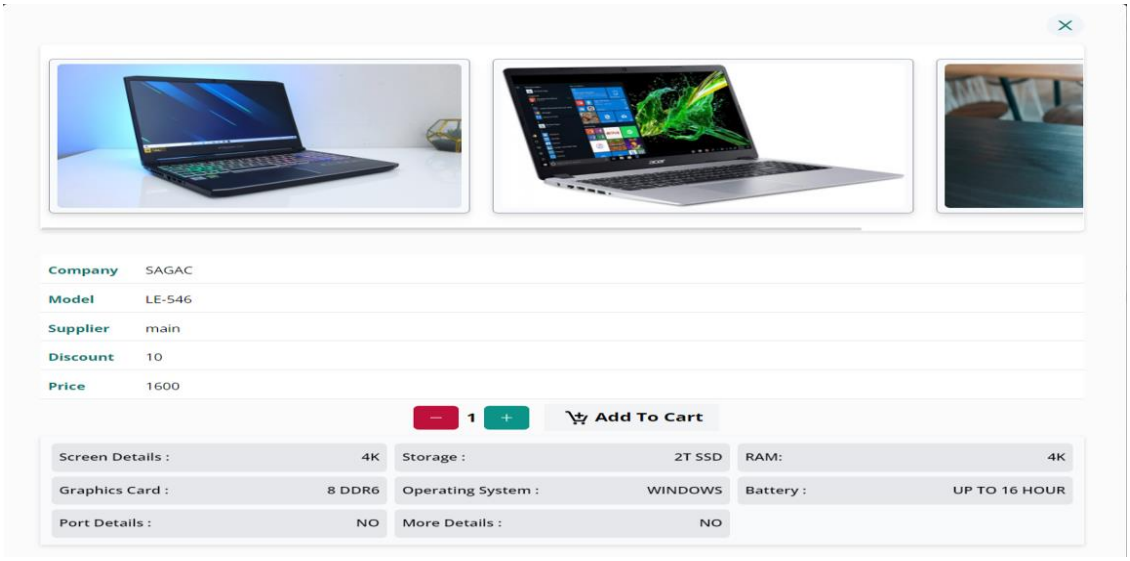

Figure 15. Screenshot of the product details page.

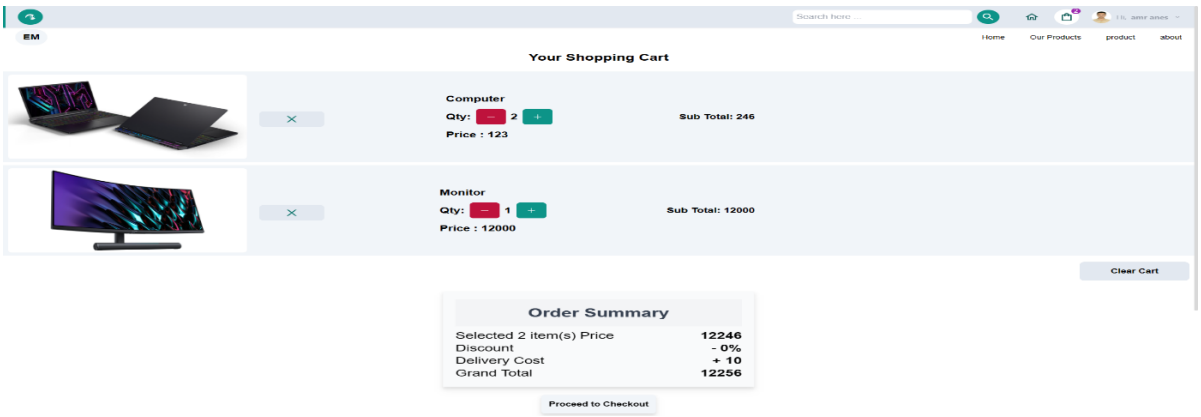

Figure 16. Screenshot of the shopping cart page.

VI. IMPLEMENTATION

Frontend Implementation

The frontend side (React JS) contains the user interface components and pages of the projects. Also, it contains the central store that store all data and states of the application to making it easy to access by all the app components.

1) Project Folder Structure (Front-End)

In the figure 17 shows the architecture of the folders of front end, which explain the sequence of components and pages in our application.

2) Redux toolkit & context

The figure 18 part of code shows the Redux store that used to store the data of the application. In the redux toolkit we can store the data in different slices to make it readable and maintainable. Redux Toolkit includes the Redux Thunk middleware, that allow to make asynchronous request and side effects within Redux actions [20], as shown in figure 19.

3) Backend Data Connectivity (AXIOS, Fetch for REST API)

The simple data fetching module for REST API connectivity using all the required methods like GET, POST, PUT, DELETE. we use Fetch API which is a modern, a more efficient and powerful approach for make HTTP requests and fetch resources, handle responses, and handling data asynchronously in a more streamlined manner in our application. in simple word we use to fetch data from API to the React JS side (store) and post data from react to the API.

Fetch API

In figure 19 show fraction of code used to fetch the all products from the API by fetchProducts function and store it in the Data state by setProducts reducer in the React JS. fetchProducts functions return functions that can perform asynchronous operations, such as API calls, and make loading states, success responses, and error handling.

Post API

In figure 20 show code used to Login. Used to search of user from the employee and customer data during the login process.

4) Internationalization (React-i18next)

Is a library that enable multilingual support for us react Js app. It provides a range of features that make it convenient to implement language switching, handle missing translations with fallback language settings, and configure translations within the app's rendering process. the language specific content is separate from the application's code, making it easier to manage and maintain localized versions.

Backend implementation

The backend side (Laravel) deals with authentication, routing, caching, validate, and store the data in the database and also handle the routes, processing the requests. In this section we will introduce some fractions of the codes that do this work.

1) Routes

Contain all URL paths or routes. Each route is connected to a specific controller which deals with data of a specific table (select, delete, insert, …). The Controller component work as mediator between the frontend and Model components. Its receive the requests or routes from the frontend, and collaborates with the model to query data or update existing data, then return the data as JSON to the frontend. In other word all operations of store, query, add, update, delete are done in controller, in simple word the requests are handled in controller. Figures 23 shows all API routes with controller associated with each one.

2) Validation

Before adding new products this fraction of code validates the data received from the frontend if the data matches the specification, it is passed. In the figure 21-22 show fraction of code for product validation and Image validation.

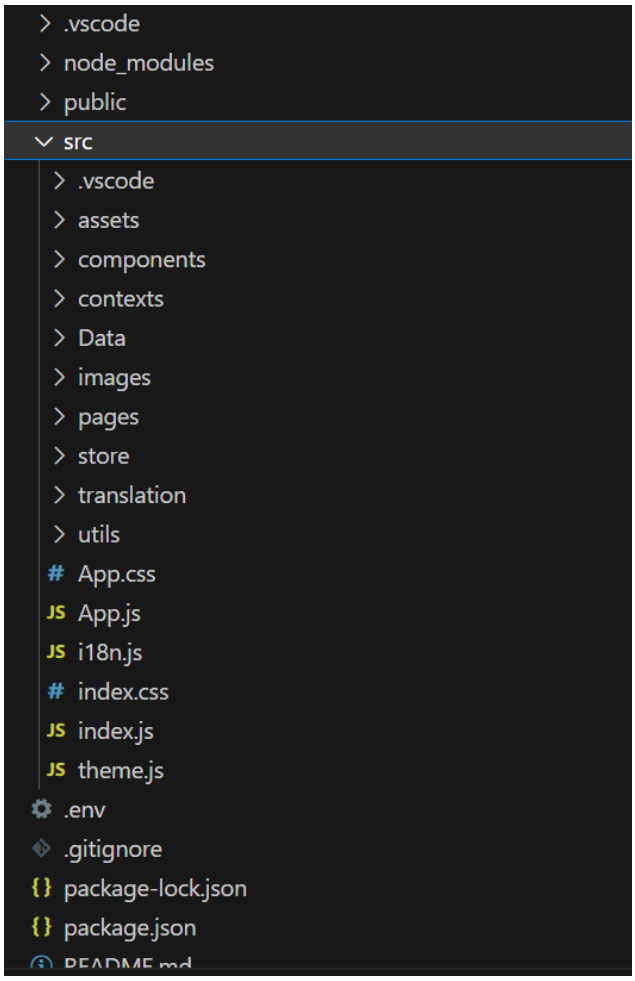

Figure 17. Screenshot of the project folder structure.

```
import { configureStore } from "@reduxjs/toolkit";
import productReducer from "./productSlice";
import latesProductReducer from "./latsProductSlice";
import categoryReducer from "./categorySlice";
import sliderReducer from "./sliderSlice";
import modalReducer from "./modalSlice";
import cartReducer from "./cartSlice";

const store = configureStore({
  reducer: {
    product: productReducer,
    latesProduct: latesProductReducer,
    sliderData: sliderReducer,
    category: categoryReducer,
    modal: modalReducer,
    cart: cartReducer,
  },
});

export default store;
```

Figure 18. Screenshot of the store

```
export const fetchProducts = () => {
  return async function fetchProductThunk(dispatch) {
    dispatch(setStatus(STATUS.LOADING));
    try {
      const response = await fetch(`${BASE_URL}products`);
      const data = await response.json();
      dispatch(setProducts(data));
      dispatch(setStatus(STATUS.IDLE));
    } catch (error) {
      dispatch(setStatus(STATUS.ERROR));
    }
  };
};
```

Figure 19. Screenshot of the fetch product

```
const handleSubmit = async (e) => {
  e.preventDefault();
  const formData = new FormData();
  formData.append("_method", "PATCH");
  formData.append("password", password);
  formData.append("user_name", userName);
  await axios
    .post(`http://localhost:8000/api/login`, formData)
    .then((response) => {
      console.log(response.data);
      // console.log(response.data.length);
      if (response.data) {
        setLogInData(response.data);
        setIsLogedIn(true);
        localStorage.setItem("logedIn", true);
        localStorage.setItem("loginInfo", JSON.stringify(response.data));
        navigator(-1);
      }
    })
    .catch((error) => {
      console.log(error);
    });
  setaPassword("");
  setUserName("");
};
```

Figure 20: Screenshot of the post login

```
public function store(Request $request)
    {
        $request->validate([
            "product_type" => "required",
            "company" => "required",
            "model" => "required",
            "price" => "required",
            "supplier" => "required",
            "video" => "required",
            "discount" => "required",
            "total_amount" => "required",
        ]);
```

Figure 21: Screenshot of the product validation

```php
public function store(Request $request)
    {
        for ($i = 0; $i < $request->numImages; $i++) {
            $imageName =
                Str::random(40) .
                "." .
                $request->{"image" . $i}->getClientOriginalExtension();
            $dataToStore = [
                "product_id" => $request->product_id,
                "image_name" => $imageName,
            ];
            Image::create($dataToStore);
            Storage::disk("public")->putFileAs(
                "products/images",
                $request->{"image" . $i},
                $imageName
            );
        }
        return response()->json([
            "message" => "done successfully",
        ]);
    }
```

Figure 22. Screenshot of the Image validation

```php
Route::get("/products/edit/{id}", [ProductController::class,"getOneProductInfoAdmin",]);
Route::get("/products/oneProduct/{id}", [ProductController::class,"getOneProductInfoCustomer",]);
Route::any("/login", [EmployeeController::class, "getOneEmployeeInfo"]);
Route::any("/employees/editOneValueForEmployee", [EmployeeController::class,"editOneValueForEmployee",]);
Route::any("/searchProducts/{searchValue}", [ProductController::class,"searchProducts",]);
Route::any("/products/filter/{type}", [ProductController::class,"getFilterProduct",]);
Route::any("/products/discount", [ProductController::class,"getProductsWithDiscount",]);
Route::any("/products/supplier/{supplier}", [ProductController::class,"getOneSupplierProducts",]);
Route::any("/orders/supplier/{supplier}", [OrderController::class,"getOneSupplierOrders",]);
Route::any("/images/edit/{id}", [ImageController::class,"updateProductImages",]);
Route::resource("images", ImageController::class);
Route::any("/customers/editOneValueForCustomer", [CustomerController::class,"editOneValueForCustomer",]);
Route::any("/statistics/year/{date}", [StatisticController::class,"getYearStatistics",]);
Route::any("/statistics/month/{date}", [StatisticController::class,"getMonthStatistics",]);
Route::any("/statistics/oneSupplier/daily", [StatisticController::class,"getDailyStatisticsForOneSupplier",]);
Route::resource("products", ProductController::class);
Route::resource("advertisements", AdvertisementController::class);
Route::resource("customers", CustomerController::class);
Route::resource("employees", EmployeeController::class);
Route::resource("orders", OrderController::class);
Route::resource("laptops", LaptopController::class);
Route::resource("modems", ModemController::class);
Route::resource("cables", CableController::class);
Route::resource("monitors", MonitorController::class);
Route::resource("watches", WatchController::class);
Route::resource("speakers", SpeakerController::class);
Route::resource("hard_disks", HardDiskController::class);
Route::resource("cell_phones", CellPhoneController::class);
Route::resource("power_banks", PowerBankController::class);
Route::resource("sliders", SliderController::class);
Route::resource("/statistics", StatisticController::class);
Route::resource("/productOrder", ProductOrderController::class);

Route::middleware("auth:sanctum")->get("/user", function (Request $request) {
    return $request->user();
});
```

Figure 23: Screenshot of the all routes for API

VII. Conclusion

We builded e-commerce application as single page application using react Js to build the front end by react components its possible for single page applications to display required content without requiring the browser to reload the entire page and the Redux as central store that store all data and states of the application to making it easy to access by all the app components. and Laravel backend framework to build a Restful API to all type of interfaces it was phone, desktop, web.

Here is a list of the Objectives that have been achieved:

- ✓ **Enhancing performance, efficiency and reliability of e-commerce platform by using ReactJS and Laravel.**
- ✓ **develop an application that is easily manageable, readable and maintainable.**
- ✓ **Develop a responsive design that is compatible with all types of devices without losing functionality or aesthetic appearance.**
- ✓ **Building a Secure Restful API with an authentication System, compatible with all kinds of interfaces it was phone, desktop, web.**

VIII. Acknowledgment

We would like to express our gratitude to Allah Almighty who granted us the opportunity, determination, courage, and patience in difficult times, as well as the strength to complete this work. Finally, we thank all the faculty members in the engineering department who imparted their knowledge to us and assisted us throughout the thesis period. In particular, we extend our sincere gratitude and full appreciation to Dr. Mogeeb Al-Hakimi, our project supervisor, for his valuable guidance and assistance successfully complete this scientific research

IX. REFERENCES

[1] Wohlgethan, Eric. "Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue. js." PhD diss., Hochschule für Angewandte Wissenschaften Hamburg, 2018.

[2] Kaluža, Marin, Krešimir Troskot, and Bernard Vukelić. "Comparison of front-end frameworks for web applications development." Zbornik Veleučilišta u Rijeci 6, no. 1, pp.261-282 (2018).

[3] Xing, YongKang, J.Huang, and Y. Lai. "Research and Analysis of the Front-end Frameworks and Libraries in E-Business Development." In Proceedings of the 2019 11th International Conference on Computer and Automation Engineering, pp. 68-72. 2019.

[4] S. Mumbaikar and P. Padiya. "Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, (3(5), pp.1-4). 2013.

[5] X. Chen, Z. Ji, Y. Fan, and Y. Zhan. "Restful API architecture based on laravel framework." In Journal of Physics: Conference Series, (vol. 910, no. 1, p. 012016). IOP Publishing, 2017.

[6] M. M. Amin, S. Widodo, A. Sutrisman, E. Cofriyanti, and A. Firdaus, RESTful Web Service as DAta Generator for Reporting of Academic Information System, FIRST (Forum in Research, Science, and Technology) pp. 1-5, 2020.

[7] M.M. Amin, A. Sutrisman, D. Stiawan, and Ermatita, "Mobile Application of Electronic Prescribing for Supporting E-Health Services", ICENIS (International Conference on Energy, Environment, Epidemiology and Information System) pp. 1-5, 2019.

[8] M. A. Jadhav, B.R. Sawant & A. Deshmukh, "Single Page Application using Angular JS"," International Journal of Computer Science and Information Technologies, 6(3): pp.2876-2879, 2015.

[9] M. S. Mikowski & J. C. Powell, Single Page Web Application, Shelter Island: Manning Publication Company.2014.

[10] N. Li, B. Zhang, The Research on Single Page Application Front-end Development on Vue, In Journal of Physics: Conference Series, (vol. 1883, no. 1, p. 012030). IOP Publishing, 2021.

[11] S. Tenzin, T. Lhamo, and T. Dorji, Design and Development of E-commerce: Web-application for Cooperative Store. International Research Journal of Engineering and Technology (IRJET), 9(2), pp.843-847. 2022.

[12] T. Nham, "Developing an E-commerce application prototype with ReactJS and Firebase." (2022).

[13] DAnh. le "E-Commercial Full Stack Web Application Development: with React, Redux, NodeJS, and MongoDB." (2023).

[14] M. Kankaala "Enhancing E-commerce with Modern web technologies." (2019).

[15] N. Mai, "E-commerce Application using MERN stack." (2020).

[16] J. Shetty, and al. et ."Review paper on web frameworks, databases and web stacks," International Research Journal of Engineering and Technology (IRJET), pp. 5734 - 5745, 2020.

[17] "JS 2022," . [Online]. Available: https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/. 2022.

[18] "Redux documentation," . [Online]. Available: https://redux.js.org/. 2023.

[19] "Laravel documents," . [Online]. Available: https://laravel.com/docs/10.x/installation#why-laravel. 2023.

[20] "Redux documentation". [Online]. Available: https://redux.js.org/tutorials/fundamentals/part-6-async-logic. Mar 6, 2023.