|  |
| --- |
|  |
| E-commerce Web Application Using MERN Stack. |
|  |

2

# ABSTRACT

Title Creating an E-commerce Web Application MERN Stack.

Year 2025

Language English

Pages 25 + 3 Appendices

The aim of this thesis project was to build an e-commerce marketplace exclusive- ly for students where students can build their entrepreneurship skills by selling their goods and services to people. This project is for Jiniarsoft, a startup soft- ware company. To build this application, we used the MERN stack, which is a popular development kit for building web applications.

The MERN stack, whose full abbreviation is MangoDB, ExpressJS, React, and NodeJS were built in the JavaScript language. The frontend was created using React, the database using MangoDB, and the server side using NodeJS and its library ExpressJS. Other associated packages and libraries were also used to build this application, like bootstrapping, React bootstrap, React-route-dom, Axios, JWT, Redux, and many more.

Keywords MERN, e-commarce, marketplace, React, MangoDB, NodeJS, ExpressJS, Bootstreap, Redux, React-router-DOM .

# ACKNOWLEDGMENTS

I would like to express my deepest gratitude to those who inspire me to com- plete my thesis work.

Firstly, I would like to thank my supervisor who gave me the feedback to improve my thesis work.

Secondly, I would thank Kamrul Islam, CEO of Jinjarsoft to supervise and solve many problems in my project. Also, I would like to thank Habibur Rahman Habib lead developer Jubic oy, for helping me to solve my project bug.

Finally, I would like to thank my wife and my family members for inspiring me to continue my thesis work in a tough time.

4

**CONTENTS** ABSTRACT ACKNOWLEDGMENTS

1. [INTRODUCTION 6](#_bookmark0)
2. [REQUIRMENT 7](#_bookmark1)
   1. [Requirement Analysis 8](#_bookmark2)
      1. [Registration system 8](#_bookmark3)
      2. [Easy and modern responsive UI 9](#_bookmark4)
      3. [Fast responding server side 9](#_bookmark5)
      4. [Scalable database 10](#_bookmark6)
   2. [Method 10](#_bookmark7)
3. [DESIGN 11](#_bookmark8)
   1. [Project Planning 11](#_bookmark9)
      1. [Project design planning 11](#_bookmark10)
      2. [Developer Environment 13](#_bookmark11)
      3. [Git version Control 14](#_bookmark12)
      4. [List of Package and Framework 14](#_bookmark13)
   2. [Implementation 16](#_bookmark14)
      1. [Front End implementation 17](#_bookmark15)
      2. [Backend implementation 20](#_bookmark16)
      3. [Database Implementation 24](#_bookmark17)
      4. [Deployment 27](#_bookmark18)
4. [RESULT 28](#_bookmark19)
5. [DISCUSSION 31](#_bookmark20)
6. [CONCLUSION 32](#_bookmark21)

[REFERENCES 33](#_bookmark22)

## LIST OF FIGURES, CODE SNIPPET AND TABLES

Figure 1: Admin page design Figure 2: Seller page design Figure 3: Buyer page design Figure 4: Project file structure Figure 5: Postman GET request. Figure 6: Home page

Figure 7: Admin Home page Figure 8: Buyer Product List Figure 9: Product list page

Code Snippet 1: Login page function Code Snippet 2: redux-user-action Code Snippet 3: redux-user-reducer

Code Snippet 4: react-bootstrap in footer Component. Code Snippet 5: Middleware for verify token.

Code Snippet 6: Token generate function. Code Snippet 7: Model of Seller

Code Snippet 8: function to store data in database. Code Snippet 9: Bcrypt hash operation

Table 1: route command table Table 2: HTTP status Table

6

# INTRODUCTION

This e-commerce marketplace is developed under the supervision of Jinnarsoft. It is a startup software company who develops web and mobile applications for local and international clients. Jinnarsoft created this project for a local client who uses this application to make a platform exclusive for university students who want to start their entrepreneurial journey.

In this project the client desired to present an e-commerce platform where only student register as seller. They can sell their products or services to another user from general people. Those general people can register as normal buyers but cannot register as seller.

The main aim for this type of platform is to reduce the dependency of students on the job market and to encourage them to start their business. It also provides students with a suitable platform where they do not have to compete with big established businesses and get suggestions for business related documentation or if necessary, get mentorship from a business organization. Also, it will bring new innovative products or services to the market and create more jobs in the future. For this the client needs an e-commerce website and Jinnersoft is devel- oping this project. Under their supervision a part of this project is developed by me.

To create this e-commerce, The MERN stack technology was selected. For the past few years it has become more and more popular in the web developer community by surprising other stack technology such as LAMP, PERN, due to its most developer-friendly & highly customizable React Admin Template, testing tools and big community support from their developer(1).

This thesis illustrates how this technology can be used to develop and to test the project. Also, the use of its associate plug in and libraries and their functionality

has been described. This paper has been divided into five chapters: are Require- ment, Design, Results, Discussion and Conclusions.

In the requirement chapter the type of criteria and the method that has been used to develop the project, the reason to select this method and analysis of each requirement are explained. The design process of the project UI, functions and libraries that has been implemented to create it and finally testing the over- all project are described in the design chapter. In the result chapter the final out- put of this project and the demonstration of the whole project view are shown. The analysis of the project output, discussion about the project limitation and possible solutions to overcome this limitation and my role and contribution to this project are described in the discussion chapter. Finally, in conclusion the summary of the whole project from beginning to end is given.

8

# REQUIREMENTS

Before starting this project work, the client gave us certain requirements for this project. Due to this, the application mainly targets a specific education communi- ty so there are some requirements that must be fulfilled. Those are:

1. Only students over 18 years old can register as seller and they need to prove their studentship.
2. An easy and modern responsive UI is needed to engage the student in the platform.
3. Fast responding server
4. Scalable database

## Requirement Analysis

After analyzing the requirements, various possible solutions to handle this re- quirement had arisen. Of those solutions some were rejected due to complexity to the user and security issue.

## Registration system

Although the application targeted mainly the student community, all types of users can use this platform for purchasing products or services. This makes the registration system much more complicated because the registration system must be developed in the way that the system can identify students and validate them.

First to validate the student, the student ID with QR code was considered to be uploaded into the system. If there is student information, then it would be easy to request the student data from their institute to verify it. But some institutes do not provide this QR coded ID card, and they also do not want to share their student data to third party organizations. Also, numerous times uploaded images might be blurry and hard to read. Therefore, after analyzing this the final deci-

sion was that the student’s email will be used as a verification method and also it will fulfill the requirement.

## Easy and Modern Responsive UI

As per the requirement, an easy method to navigate and modern user interface for the project was needed. Since most of the targeted audience are young stu- dents. Also, the project had to be made responsive to all devices such as desk- top, mobile and tablet.

To fulfill this requirement, the front-end of the project was focused on. Some frameworks had been searched where the project could get the most customiza- ble lay out. Also, the type of framework that has more community support on the internet was chosen.

After considering all these factors the React framework was chosen for improv- ing the UI of the application. It is free and open source. Because it is developed by Meta, formerly known as Facebook, new updates are always released by their company developer (2). Also, because it is open source, it can get lots of support from developer communities all around the world. It is a JavaScript library mainly used for front end development.

React also utilizes another framework Bootstrap which is a CSS framework and also free, open source enrich with plenty of customizable layouts. Also, it has community supports on the internet (3).

## Fast Responding Server Side

To run all operations smoothly in backend, the client asked to build a fast re- sponding server side. There are several technologies such as Laravel, Django, Ru- by on Rails and Express. But among all expresses, the framework of Node js is fastest (4).

10

Node itself is an open-source and cross-platform JavaScript runtime environ- ment. It is used to write JavaScript in command line for server-side scripting. With Express, which is a framework of Node js, it can put much more functionali- ty by using its build in function like “**auth**” “**error-pages**” in the application back end (5).

## Scalable database

To provide a scalable database to the client’s project, the MangoDB was chosen. Aside from scalability, MangoDB is much more user friendly and flexible than MySQL (6). In the project, many users may upload their data for example an im- age or file, in this case MangoDB has InsertMany() API for rapidly inserting data, prioritizing speed over transaction safety wherein MySQL data needs to be in- serted row by row(7).

## Method

After analyzing the entire requirement for the project, it was decided to develop the entire project with the MERN stack method. Because it uses uniform pro- gramming language JavaScript, our project was simple and fast.

# DESIGN

In this chapter, the entire development process from start to finish is elaborated. This chapter first and foremost discusses pre planning such as designing the page, IDE selection, version control. Secondly, the implementation such as front end logic, backend logic, database logic is discussed.

## Project Planning

Before starting any project, it is mandatory to plan its UI. In this case, the project must have three types of user categories. One is an admin who controls the en- tire project, a seller or a student who sell their products and a buyer or a normal user who can see the products and order the product from the site.

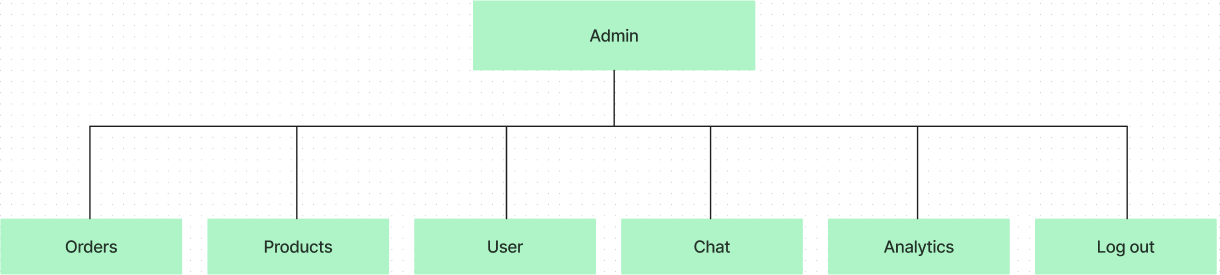
## Project Design Planning

The admin, seller, and buyer each will have their own accounts and each account has their restrictions except the admin who will control the site.

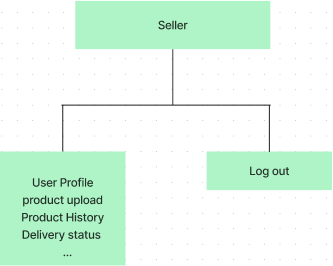
Initially the admin only uploads the product so the admin had the order page where they can monitor the order, products to create a product and edit a prod- uct, the user page to monitor or remove the user, chats pages used to check the message from the user, analytics is to check the sell graph of every month and log out is to logout from the account.

After completing registration in the system, sellers or students can also sell their products. When sellers log in their account, they will see their profile where they can upload their products and see the history of their products.

12

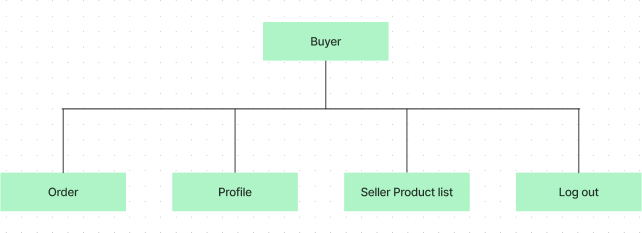


**Figure 1:** Admin page design



**Figure 2:** Seller page design

For buyers or general users if they log in, they are directed to their user profile. They also have a product page where they can see their order history. If Buyers update their profile page they can see the buyer’s products exclusively.



**Figure 3:** Buyer page design

## Developer Environment

The right selection of development environment makes the entire project devel- opment process smoother. For web application there several IDEs right now. But for this project, Visual Studio code is selected. The main reason for choosing is that it is easy to navigate and there are plenty of extensions which helped to ex- ecute the code more smoothly. The list of extension of VS code that has been used is given below.

* + - 1. ES7+ React/Redux/React-Native snippets: Extensions for React, React- Native and Redux in JS/TS with ES7+ syntax. Customizable. Built-in inte- gration with prettier font.
      2. Babel JavaScript: VSCode syntax highlighting for JavaScript.
      3. Live Preview: Hosts a local server in workspace to preview the webpages on.
      4. vscode-icons: Icons for Visual Studio Code

14

## Git version Control

To collaborate in a project with other developers, a version control system was required. There are many versions of control systems such as Git, AWS Co- deCommit and Azure DevOps. In this project the Git version system was used.

In this project Git was initially installed by using command **“git init”**. Then the status of code was checked using code **“git status”**. Then the code using code **“git commit”** and **“git push”** was committed and pushed**.** At the time of Git commit the version name was mentioned so that the next time whenever the previous code is needed, it could be retrained. The Git cloud service called “**github**” was used which is one of the popular Git cloud services.

## List of Package and Framework

For the backend Node js was used in this project. Some necessary node packages needed to be installed for the add backend and frontend functionality. The list of Node package is given below.

* + - 1. Install npm version 16.17.1. A library and registry for JavaScript software packages. For package management the npm was needed in the pro- ject(8).
      2. Install npx version 8.15.0. A npm package runner which allows run some npm package without even installing it(9).
      3. Install react version 17.0.2. A JavaScript library for building user interfac- es.
      4. Install bootstrap version 5.1.3. A front-end framework for developing re- sponsive, mobile first projects.
      5. Install react-bootstrap version 2.2.1. A Bootstrap 5 components built with React.
      6. Install react-dom version 17.0.2. It serves as the entry point to the DOM and server renderers for React(10).
      7. Install react-router-dom version 6.2.2. It contains bindings for using [React](https://github.com/remix-run/react-router) [Router](https://github.com/remix-run/react-router) in web applications.
      8. Install redux version 4.1.2. One of the most important package because helps to write applications that behave consistently, run in different envi- ronments (client, server, and native), and are easy to test(11).
      9. Install mongoose version 6.2.8. A [MongoDB](https://www.mongodb.org/) object modelling tool de- signed to work in an asynchronous environment.
      10. Install multer version 1.4.5. For uploading files.
      11. Install nodemon version 2.0.15. To develop Node.js based applications by automatically restarting the node application when file changes in the di- rectory are detected(12).
      12. Install nodemailer version 6.9.1. To help to send mail by node application. In this case we use this package to send OPT to user email(13).
      13. Install jsonwebtoken version 8.5.1. To create web token for login pur- pose.
      14. Install dotenv version 16.0.3. To load environment variables from a .env file into process.env.
      15. Install bcryptjs version 2.4.3. To help to create hash passwords to im- prove security(14).
      16. Install Axios version 0.26.1. To provide Promise based HTTP client (PUT, GET, DELETE) for the browser and Node.js(15).
      17. Install helmet version 5.1.1. To provide security for Express apps by set- ting various HTTP headers.

Several other packages were used to create the project, but the ones above are most important. All the names of the packages and their versions can be found in the package. json file in the project folder.

16

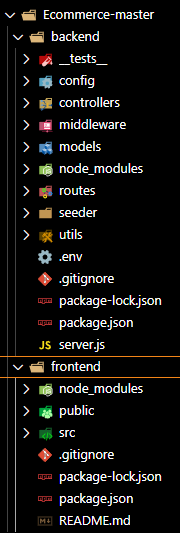
## Implementation

In this section the implementation of the code is discussed. The whole project file is divided mainly between frontend and backend part. In the frontend the UI of the project and in the backend the functionality of the project have been de- veloped.

The frontend is divided into several file that are public, for example src or node module. The src file is the most important because all the page components are in there. In the public file the default image has been stored in the project slider and an uploaded image from the user will be stored there. The Node module is an auto generated folder and package json is also auto generated where all the frontend packages are displayed. The details about frontend are discussed in sec- tion 3.2.1.

The backend also divided into many folders and the most important are config. This one connects the database and gives us status about the connection. con- troller folder control all the operation of the pages, middleware used to the login operation, model declares all the pages required value, route have route instruc- tion of the pages, seeder folder have demo data, utils have all the necessary function commend. The details about the backend operation are discussed in

3.2.2 section.



**Figure 4:** Project file structure

## Front End Implementation

In this project the frontend is divided into three sperate users (admin, seller, buyer) and separate components were created for each of them to make it sim- pler. All the pages and their components and utils are stored in the “src” folder. But because of the huge number of components, the state management package Redux was used and configured for global page management. The “Login page” is discussed in detail so that readers can understand the Redux application in pro- ject components.

For simplicity a login page and login page component were created separately. In the log in page, the component contains all the Bootstrap elements, objects, and condition and from the login page when the user signs up and logs into their ac- count and “loginUserApiRequest” function is created and POST the data to

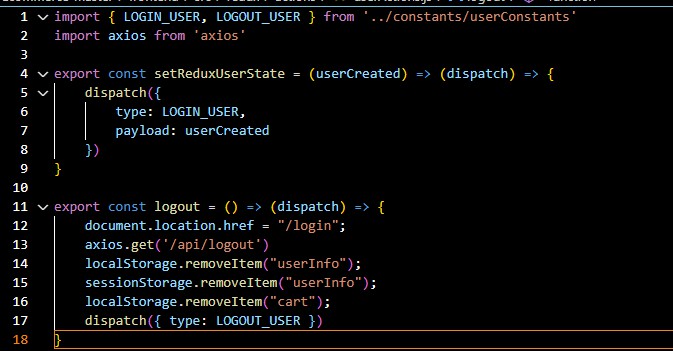
18

/api/user/login. By this backend folder the function gets the email and password and if the validation is successful the token with data is stored into “userinfo” and if the process fails, then an error massage is passed in payload and takes the user to a corresponding route which is the log in page and shows an error mes- sage in there.

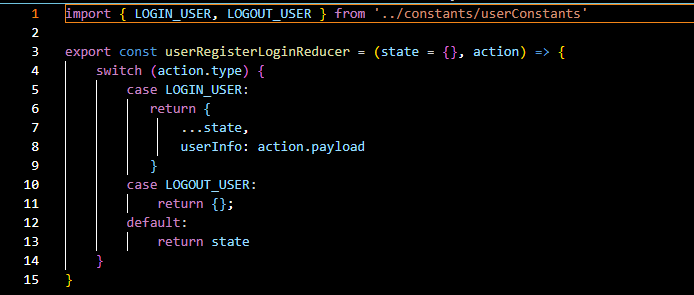


**Code Snippet 1:** Login page function

To stay logged in in the user profile all the time until user logs out “Redux” was used. A constant variable in the Redux constant folder was created. Then this constant was passed to the action and reducer file. In the reducer file the state of the component changes according to the action. If “userinfo” changes due to get api/logout then action will be changed and the reducer will also change the state immediately, otherwise it keeps the same state all the time.



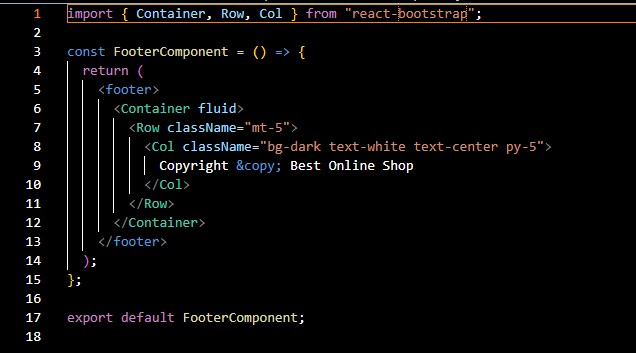
**Code Snippet 2:** Redux-user-action



**Code Snippet 3:** Redux-user-reducer

20

The Bootstrap and the React-bootstrap package were also used to create the UI. With Bootstrap the header and footer layout were created using classes such as Row, Col, Carousal.



**Code Snippet 4:** React-bootstrap in Footer Component.

## Backend Implementation

The backend was created with Node js and its framework Express js. To establish client and server communication the HTTP communication protocol was created. Mainly four protocols were used to create communication.

The route of a page is being called by using this request. Each route serves a dif- ferent purpose, and each route can be accessed by a particular user. Some of the routes and corresponding pages activity is listed below in Table 1.

**Table 1:** Route command table

|  |  |  |
| --- | --- | --- |
| **Route** | **Purpose** | **Access** |
| router.get("/", getUsers) | Get user information | Admin |
| router.delete('/:id', deleteUser) | Delete user | Admin |
| router.get("/search/:searchQuery", getProducts) | Search product | Public |
| router.post("/seller/product", create- Product); | Product creation | Seller |
| router.patch("/seller/setSold", setSold); | Mark as sold | Seller |
| router.put("/delivered/:id", up- dateOrderToDelivered); | Update delivery | Admin |
| router.get("/admin", getOrders); | Check the order | Admin |

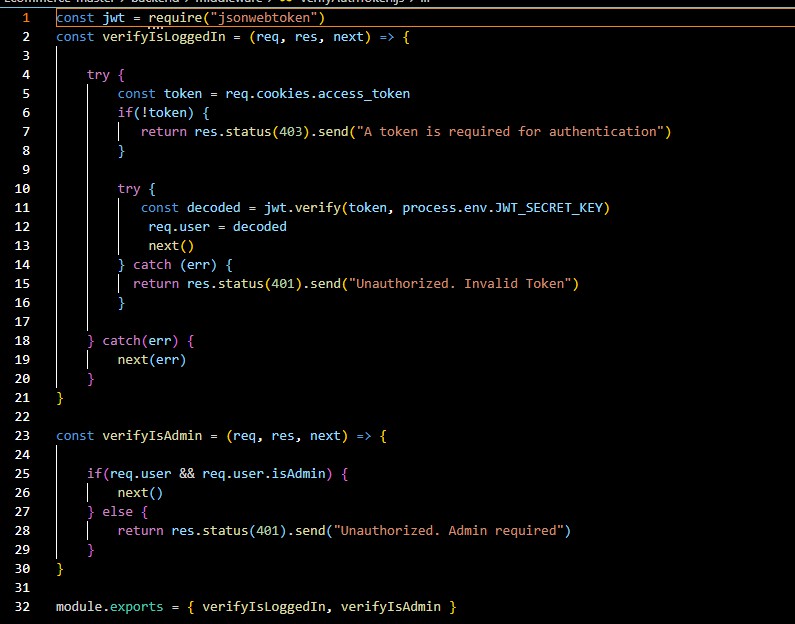
Specific pages are assigned with a controller. Each has different functionality and condition(16), depending on the condition response status change. Some re- sponse statuses are shown below in Table 2.

**Table 2:** HTTP status Table

|  |  |
| --- | --- |
| **Status** | **Function** |
| res.status(400) | Bad Request |
| res.status(201) | Created |
| res.status(500) | Internal Server Error |

22

With this the authentication for different users was created. Middleware “JSON web token” was used for this authentication. If a visitor wants to log in protected by jwt protocol, it will create an access token and send it to database. If the to- ken match with the jwt secret key is set by the admin, it will give a desired access token and the user can be logged in.



**Code Snippet 5:** Middleware for verify token.



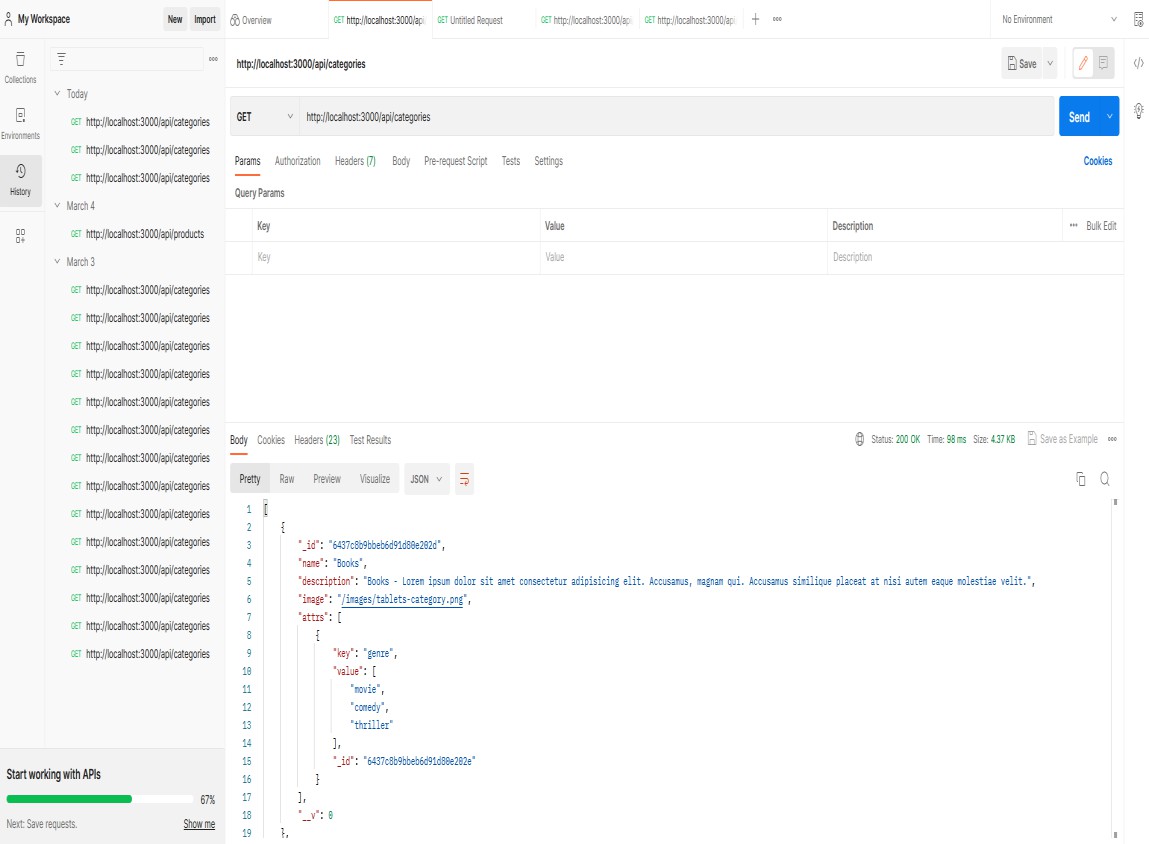
**Code Snippet 6:** Token generate function

The POSTMAN application has been used to test the api before calling in front end(17). Here, <http://localhost:5000/api/categories> were used to get a request to see the response from the api. Here local host 5000 was used because the ap- plication backend runs in port 5000.

With this request JSON format data can be obtained from the database. It’s a

very handy tool when the api is tested.

24



**Figure 5:** Postman GET request.

## Database Implementation

From the database MangoDB and its object model Mongoose was used to create the database section. A model was created to define the architecture for specific pages in the data base. Generate-schema was used to convert JSON Objects to MySQL Table Schema. It gives a better format in the database.

When someone creates an MangoDB account then an environment key is also created for database and it puts in an .env file. Also, the api secrete key and URL can be obtained which is also used to connect the database with the application. When the data is being pushed in the database, a unique id with all the data and date is uploaded in the database.



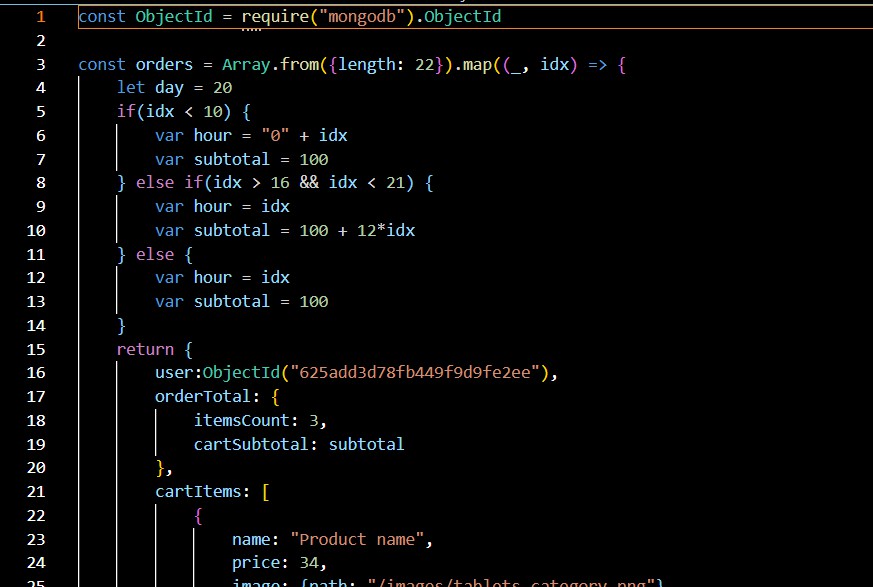
**Code Snippet 7:** Model of Seller

Using this ID, the seeder dami data is being pushed in the database. It helps to test if the demo data is stored in the database or not.

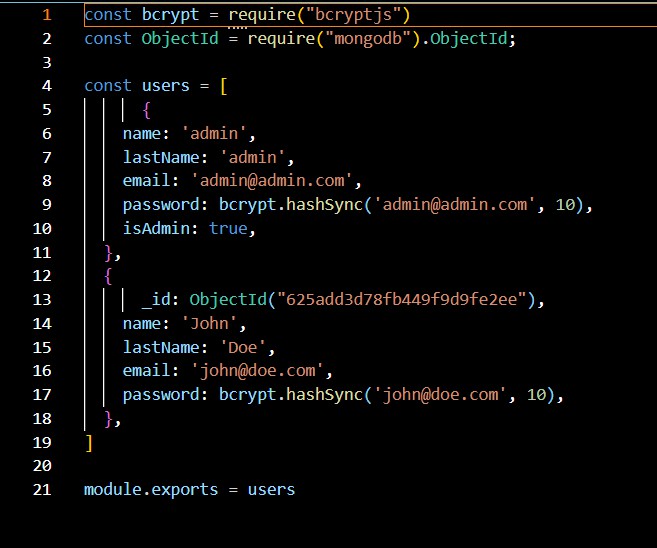
The bcrypt package is used to manage to encrypt the original password. This pro- cess is called hashed algorithm which changes the password in a hashed format so other people cannot read it.

A config file is also created which shows the connection status. An async function is used because everything is asynchronous when it comes to the data base.

26



**Code Snippet 8:** Function to Store Data in Database



**Code Snippet 9:** Bcrypt hash Operation

## Deployment

For deployment “Render” was used. This is a cloud service where developers can build and run their applications (18).

It is very easy to deploy the application in this site. A new “gitignor” file was cre- ated and “.env” was written. Then a “personal access token” had to be created in Github and the entire project uploaded in the Github repository.

After that the Github account was linked in Render and a live server in the render application was created. After putting necessary credential, the project was de- ployed in the live server using a unique link created by Render.

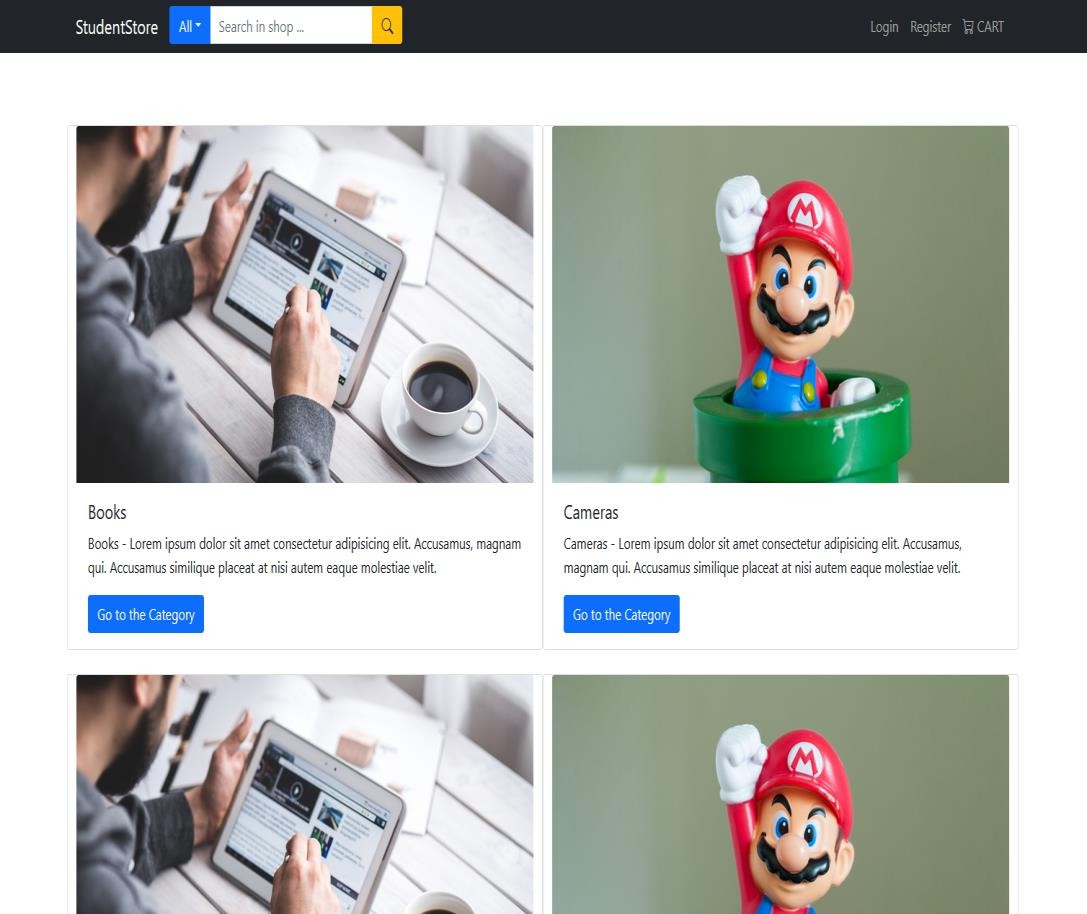
The link of the live server is given below.

Back end live server: https://backend-wyqt.onrender.com Front End live server: https://frontend-o6mi.onrender.com/

28

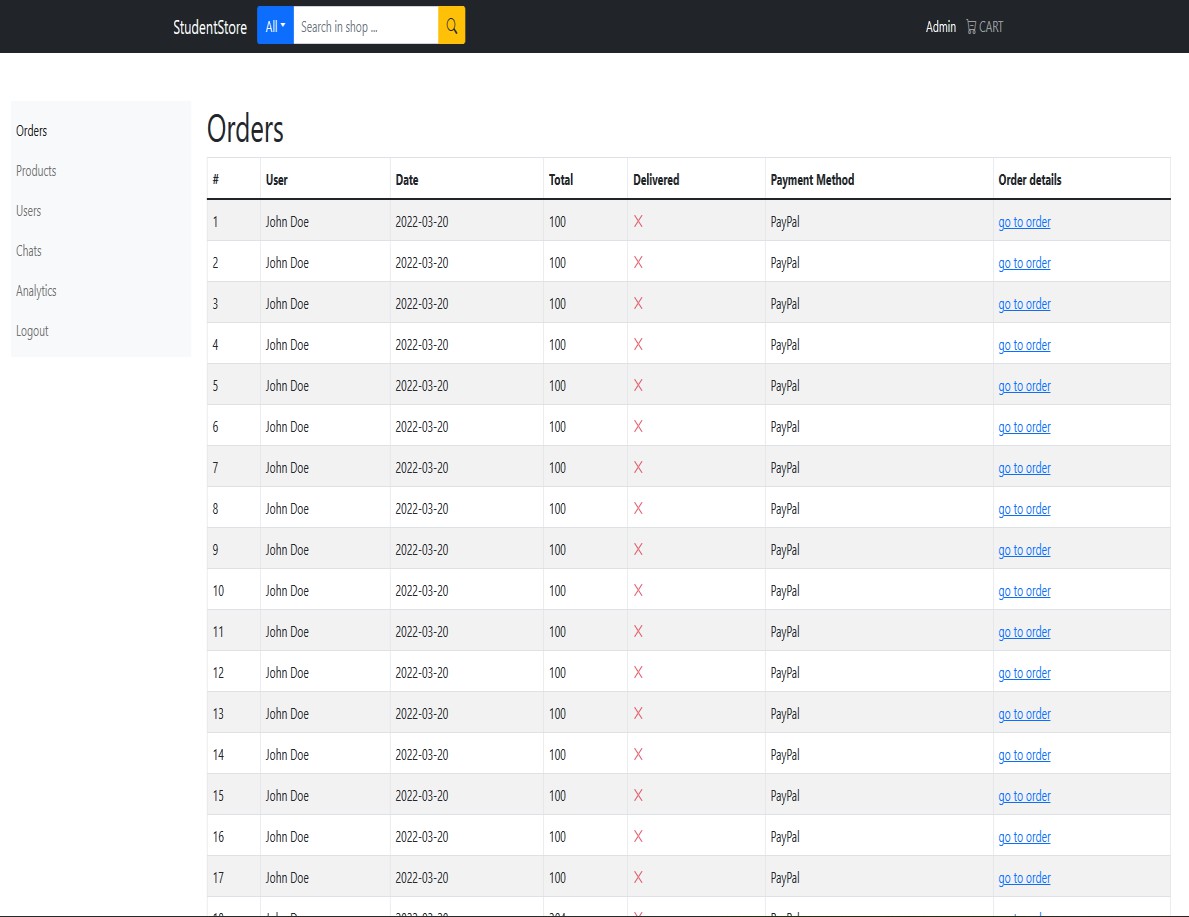
# RESULT

In this chapter the project is demonstrated and described.



**Figure 6:** Home page

The home page (Figure 6) displays all the products of the seller and buyer. It also shows the login and registration and cart option.



**Figure 7:** Admin Home page

When the admin logs in (Figure 7), then they will see the several options: order list, product, user, chat, analytics and log out. In the order list the admin can see all orders from the site, in the product menu the admin can upload the product, in the user option the admin can create and delete a user, in the chat option the admin can see the message of the user, in the analytic option the admin can view the buy and sell cart and the logout option is for log out.

30

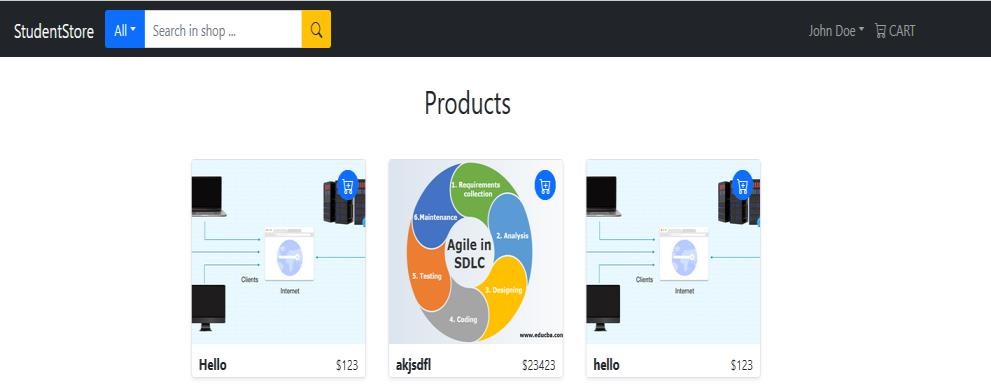


Figure 8: Buyer Product List

In Buyers profile page (Figure 8) the buyer or user can see the product from the e-commerce site. From there the user can order the product and it goes to the cart.

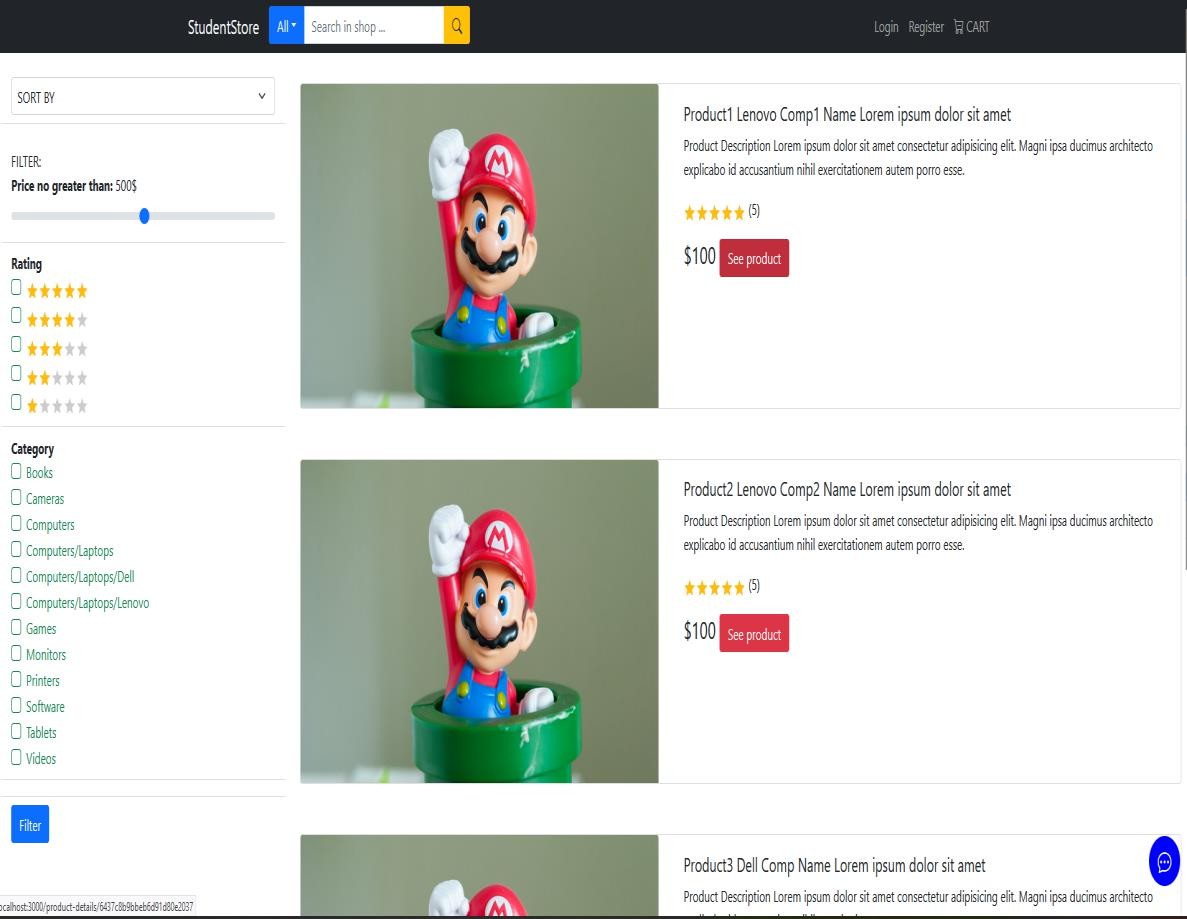


Figure 9: Product list page

In the Product list page (Figure 9) the user can sort the product base on rating and category.

# DISCUSSION

This chapter discusses what kind of problem this project has solved and what can be learned through this process.

This project contributes mainly to front end part but some backend part work was also done. Some UI and implementation using React was designed. Log in and register page UI were created and its back-end work also done. Controller, route for user, seller and buyer were created.

From this work one can learn a lot about React package management and how a state works. By doing backend work the understanding of server-side work can be improved. One can learn how to connect front and backend in a project and what kind of package would be needed to accomplish this work.

Due to lack of time some functionality is not operational for example the, cart page. In the live server the buyer profile page is not appearing. In future this problem can be solved and this application can be published for commercial use.

There are some limitations of the project, first the authentication can be much more sophisticated. Checking the validity of the student should be developed further. A more colorful and easier UI for the application is also needed.

32

# CONCLUSIONS

This project is an innovative project. The aim of this project is to create a youth entrepreneur from the university. For that a fast, easy, and modern application is required.

The MERN method has been used to complete the project successfully because it is faster than other web development stacks. In the project React was used to develop the UI and Node was used to develop the backend.

Easier UI can make this project better for the user.

Finally, this project brings a new idea and with new stack technology like MERN it can be the next hit web application on the internet.

# REFERENCES

1. Best Web Development Stack 2023 Accessed 23.3.2023 <https://dev.to/theme_selection/best-web-development-stack-2jpe>
2. React official page Accessed 23.3.2023 <https://legacy.reactjs.org/>
3. The Bootstrap Blog Accessed 23.3.2023 <https://blog.getbootstrap.com/>
4. Express official page Accessed 23.3.2023 <https://expressjs.com/>
5. Express examples Accessed 23.3.2023 <https://expressjs.com/en/starter/examples.html>
6. MangoDb official page Accessed 23.3.2023 <https://www.mongodb.com/>
7. MangoDb resources Accessed 23.3.2023 [https://www.mongodb.com/docs/manual/reference/method/db.collection.i](https://www.mongodb.com/docs/manual/reference/method/db.collection.insertMany/) [nsertMany/](https://www.mongodb.com/docs/manual/reference/method/db.collection.insertMany/)
8. npm package description Accessed 24.3.2023 <https://www.npmjs.com/package/npm>
9. npx package description Accessed 24.3.2023 <https://www.npmjs.com/package/npx>
10. react dom package description Accessed 24.3.2023 <https://www.npmjs.com/package/react-dom>
11. redux package description Accessed 24.3.2023 <https://www.npmjs.com/package/redux>
12. nodemon package description Accessed 24.3.2023 <https://www.npmjs.com/package/nodemon>
13. nodemailer package description Accessed 24.3.2023 <https://www.npmjs.com/package/nodemailer>
14. bcryptjs package description Accessed 24.3.2023 <https://www.npmjs.com/package/bcryptjs>

34

1. axios package description Accessed 24.3.2023 <https://www.npmjs.com/package/axios>
2. HTTP response status codes Accessed 25.3.2023 <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>
3. Postman starter page Accessed 26.3.2023 [https://www.postman.com/postman/workspace/postman-public-](https://www.postman.com/postman/workspace/postman-public-workspace/documentation/12959542-c8142d51-e97c-46b6-bd77-52bb66712c9a) [workspace/documentation/12959542-c8142d51-e97c-46b6-bd77-](https://www.postman.com/postman/workspace/postman-public-workspace/documentation/12959542-c8142d51-e97c-46b6-bd77-52bb66712c9a) [52bb66712c9a](https://www.postman.com/postman/workspace/postman-public-workspace/documentation/12959542-c8142d51-e97c-46b6-bd77-52bb66712c9a)
4. Render offcial page Accessed 26.3.2023 <https://render.com/>