|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| VILNIAUS KOLEGIJA  UNIVERSITY OF APPLIED SCIENCES  FACULTY OF ELECTRONICS AND INFORMATICS  Image result for viko logo | | |  | | | VILNIUS COLLEGE  Image result for viko logoFACULTY OF ELECTRONICS AND INFORMATICS |
|  | | |  | | |  |
| **INTERNET OF THINGS** | | |  | | | **INTRODUCTION TO INFORMATICS** |
| PRACTICAL ASSIGNMENT  INTERNET OF THINGS FOR INVENTORY MANAGEMENT  6531BX028 PI18E | | |  | | | PRACTICAL ASSIGNMENT  SPOTIFY USER MANUAL  6531BX028 PI18E |
| STUDENT | DŽIUGAS PEČIULEVIČIUS | STUDENT | | DŽIUGAS PEČIULEVIČIUS |
| (SIGNATURE)  3/16/2021 | | |  | | | LECTURER |
| LECTURER | AIRINA SAVICKAITĖ | (SIGNATURE)  10/17/2018 | | VIRGILIJUS KUKLIERIUS |
| (SIGNATURE)  3/16/2021 | | |  | | | 2018 |

2021

**TABLE OF CONTENTS**

[INTRODUCTION 4](#_Toc67397904)

[PURPOSE 5](#_Toc67397905)

[1. HOW CAN THIS BE HELPFUL IN ECOMMERCE? 6](#_Toc67397906)

[**FUNCTIONAL AND NON-FUNCTIONAL SOFWARE REQUIREMENTS** 7](#_Toc67397907)

[**Functional requirements** 7](#_Toc67397908)

[**Non-Functional requirements** 11](#_Toc67397909)

[**SOFTWARE DESIGN MODEL** 12](#_Toc67397910)

[**Functional requirements traceability table** 12](#_Toc67397911)

[**Entity-relationship diagram** 13](#_Toc67397912)

[**Structure diagram** 14](#_Toc67397913)

[**IoT SCANNER IMPLEMENTATION INTO THE PROJECT** 15](#_Toc67397914)

[**How would this work?** 15](#_Toc67397915)

[**SOFTWARE IMPLEMENTATION TOOLS** 19](#_Toc67397916)

[**USER DOCUMENTATION** 20](#_Toc67397917)

[**CONCLUSIONS** 21](#_Toc67397918)

[**LITERATURE** 22](#_Toc67397919)

**TABLE OF FIGURES**

[Figure 1 - Use-case diagram for functional requirements 7](#_Toc67397864)

[Figure 2 - Use-case diagram for functional requirements 12](#_Toc67397865)

[Figure 3 - Requirements traceability table 13](#_Toc67397866)

[Figure 4 - Entity-relationship diagram 13](#_Toc67397867)

[Figure 5 - Structure diagram 14](#_Toc67397868)

[Figure 6 - QR code information 15](#_Toc67397869)

[Figure 7 - Object product in the database 16](#_Toc67397870)

[Figure 8 - Program screen 17](#_Toc67397871)

# INTRODUCTION

The rise of sale in e-commerce (especially during this pandemic) is good news when it comes to revenue reports and a challenge for warehouse managers – as retailers scale and expand, keeping track of the inventory can get hard. To meet the needs of an on-demand shoppers, warehouses turn to IoT applications. In the last decades, using Internet of Things for warehouse management became more than a promising concept – companies started implementing codes, sensors, RFID tags, and other forms of connectivity to manage daily tasks.

# PURPOSE

The goals for this assignment are to know how inventory management systems can be helpful in eCommerce.

* What features does ecommerce inventory management system provide?
* To get familiar with IoT possibilities in eCommerce inventory management.
* To get to know what kind of tools can be used to manage inventory.
* To try and show some sort of example on how it works and how it can be implemented in our project.

# HOW CAN THIS BE HELPFUL IN ECOMMERCE?

First, what is E-commerce?

Ecommerce is known as electronic or internet commerce. It refers to the buying and selling goods or services using the internet, and the transfer of money and data to execute these transactions. Ecommerce is often used to refer to the sale of physical products online, but it can also describe any kind of commercial transaction that is facilitated through the internet. Whereas e-business refers to all aspects of operating an online business, ecommerce refers specifically to the transaction of goods and services [1].

Basically, the e-commerce is all about shopping online and transfering money from one end to another. There are six types of e-commerce:

* Business-to-Business (B2B)
* Business-to-Consumer (B2C)
* Consumer-to-Consumer (C2C)
* Consumer-to-Business (C2B)
* Business-to-Administration (B2A)
* Consumer-to-Administration (C2A)

Now how can inventory management systems be helpful in eCommerce?

A good inventory management strategy improves the accuracy of inventory orders. Proper inventory management helps you figure out exactly how much inventory you need to have on-hand. This will help prevent product shortages and allow you to keep just enough inventory without having too much in the warehouse [2].

Using IoT approach in inventory management these days have many benefits: automation of inventory tracking and reporting, visibility into the inventory items quantity, location and movements, inventory & time optimization.

**FUNCTIONAL AND NON-FUNCTIONAL SOFWARE REQUIREMENTS**

**Functional requirements**

Diagram

Description automatically generated

Figure 1 - Use-case diagram for functional requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **FUNCTIONAL REQUIREMENT** | **USER REQUIREMENT** | **USE-CASE** | **PRIORITY** |
| FR1, FR 7 | View | User, Admin | UC01 | 1 |
| FR2 | Add Item to the Cart | User | UC02 | 2 |
| FR3 | Login | User | UC03 | 3 |
| FR4 | Buy Product | User | UC04 | 4 |
| FR5 | Search Products | User | UC05 | 11 |
| FR6 | Write Reviews | User | UC06 | 10 |
| FR7 | Login | Admin | UC | 5 |
| FR8 | View Orders | Admin | UC07 | 6 |
| FR9 | Update Order | Admin | UC08 | 7 |
| FR10 | Edit/Delete Products | Admin | UC09 | 8 |
| FR11 | Add Products | Admin | UC10 | 9 |

Make sure that the functional requirements are formulated correctly: they are easy to understand, clear, unambiguous, implementable, verifiable.

Describe the use cases.

|  |  |
| --- | --- |
| **Function** | View |
| **ID** | UC01 |
| **Description** | User/Admin can look at the products |
| **User roles** | User, Admin |
| **Prie-conditions** | There is an internet connection |
| **Inputs** | Click View Products |
| **Outputs** | Product List |
| **Action** | 1. The user enters the website 2. User click Browse Products |
| **Post-conditions** | After clicking View Products the user should be able to see the products. |
| **Side effects** | None. |

|  |  |
| --- | --- |
| **Function** | Add Item to the Cart |
| **ID** | UC02 |
| **Description** | Customer can put item into the shopping cart |
| **User roles** | User |
| **Prie-conditions** | There is an internet connection |
| **Inputs** | The product that customer desires. |
| **Outputs** | Desired product in the shopping cart. |
| **Action** | 1. The user enters website.  2. The user click Browse Products.  3. The user click the desired product.  4. In the Product Description user click Add item to the Cart.  5. The product appears in the cart. |
| **Post-conditions** | The product appears in the cart. |
| **Side effects** | None. |

|  |  |
| --- | --- |
| **Function** | Login |
| **ID** | UC03 |
| **Description** | Customer can login into the system |
| **User roles** | User |
| **Prie-conditions** | There is an internet connection |
| **Inputs** | Email and Password |
| **Outputs** | User succesfully logins in the system. |
| **Action** | 1. The user enters website.  2. The user click Login.  3. The user click enters their Email and Password.  4. User sees that sees succesfully log in. |
| **Post-conditions** | User is succesfuly log in |
| **Side effects** | None. |

|  |  |
| --- | --- |
| **Function** | Buy Product |
| **ID** | UC04 |
| **Description** | Customer can buy the desired product |
| **User roles** | Registered User |
| **Prie-conditions** | There is an internet connection |
| **Inputs** | Address, Bank information. |
| **Outputs** | Confirmation of the order window. |
| **Action** | 1. The user checks out item in the shopping cart.  2. The user enter their customers information.  3. The user click buy.  4. The user gets a window of order confirmation.  5. User‘s Order entered in the database. |
| **Post-conditions** | The user‘s order is entered into database |
| **Side effects** | None. |

|  |  |
| --- | --- |
| **Function** | Search Product |
| **ID** | UC05 |
| **Description** | Customer can search for the desired Product |
| **User roles** | User |
| **Prie-conditions** | There is an internet connection |
| **Inputs** | Keyword |
| **Outputs** | List of product of related products |
| **Action** | 1. The user enters website.  2. The user click the search bar.  3. The user enters their keyword.  4. The system shows a products related to the keyword. |
| **Post-conditions** | System shows products related to the keyword |
| **Side effects** | None. |

|  |  |
| --- | --- |
| **Function** | Write Reviews |
| **ID** | UC06 |
| **Description** | Customer can leave reviews |
| **User roles** | Registered user |
| **Prie-conditions** | There is an internet connection, The user should be logged in. |
| **Inputs** | Message that user wants to leave, star rating |
| **Outputs** | Message appears in the reviews section. |
| **Action** | 1. User enters the website 2. User clicks Browse products. 3. User click on a chosen product. 4. User click to write review. 5. User leaves message and star rating. 6. The review appears in the product page. |
| **Post-conditions** | Review appears in the review section of the product |
| **Side effects** | None. |

|  |  |
| --- | --- |
| **Function** | View Orders |
| **ID** | UC07 |
| **Description** | Admin can view orders |
| **User roles** | Admin |
| **Prie-conditions** | There is an internet connection |
| **Inputs** | Click on orders page |
| **Outputs** | System outputs orders |
| **Action** | 1. Admin enter the website. 2. Admin logs-in 3. Admin clicks on orders 4. All orders appear on the screen |
| **Post-conditions** |  |
| **Side effects** | None. |

|  |  |
| --- | --- |
| **Function** | Update Order Status |
| **ID** | UC08 |
| **Description** | Admin can update order status |
| **User roles** | Admin |
| **Prie-conditions** | There is an internet connection |
| **Inputs** | Click on specific order |
| **Outputs** | Show order information |
| **Action** | 1. Admin enter the website. 2. Admin logs-in 3. Admin clicks on orders 4. All orders appear on the screen 5. Click on the order details 6. Click on mark as delivered. |
| **Post-conditions** | The system now shows that the order is delivered. |
| **Side effects** | None. |

|  |  |
| --- | --- |
| **Function** | Edit/Delete Product listings |
| **ID** | UC09 |
| **Description** | Admin can edit or delete the product. |
| **User roles** | Admin |
| **Prie-conditions** | There is an internet connection |
| **Inputs** | Click on products page |
| **Outputs** |  |
| **Action** | 1. Admin enter the website. 2. Admin logs-in 3. Admin clicks on products 4. All existing products appear on the page 5. Select the delete or edit button on the product |
| **Post-conditions** | The admin is able to edit or delete the product |
| **Side effects** | None. |

|  |  |
| --- | --- |
| **Function** | Add Product |
| **ID** | UC10 |
| **Description** | Admin can add product to the system. |
| **User roles** | Admin |
| **Prie-conditions** | There is an internet connection |
| **Inputs** | Product information : Product name, Description,Quantity,Price. |
| **Outputs** |  |
| **Action** | 1. Admin enter the website. 2. Admin logs-in 3. Admin clicks on products 4. All existing products appear on the page 5. Admin clicks „create product“ 6. Enters product information |
| **Post-conditions** | The admin is able to add product to the database. |
| **Side effects** | None. |

**Non-Functional requirements**

1. Application should be written in JavaScript.
2. Application should use React framework.
3. The application should be hosted on Heroku.
4. The application (shop) data should be stored on Firebase or MongoDB.
5. The system should allow users to pay for their goods using Stripe or PayPal.

**SOFTWARE DESIGN MODEL**

**Functional requirements traceability table**

Functional Requirements:

1. The system should allow users to view product collections.
2. The system should allow users to add item to shopping cart.
3. The system should have Sign-in window where user can login.
4. The system should allow users to check out and buy an item.
5. The system should allow users to search for the product.
6. The system should allow users to read and write reviews.
7. Admin should be able to view products in the system.
8. Admin should be able to view orders in the system.
9. Admin should be able to update the order.
10. Admin should be able to edit/delete products.
11. Admin should be able to add products into the system.

Diagram

Description automatically generated

Figure 2 - Use-case diagram for functional requirements

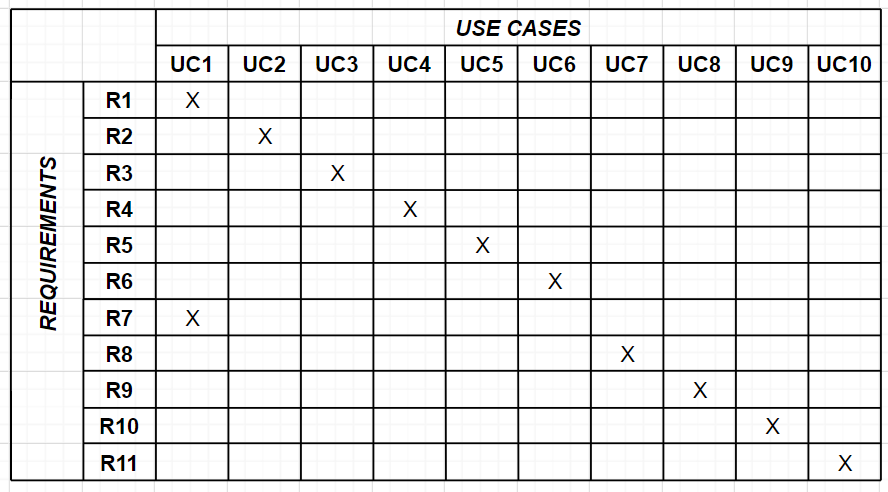


Figure 3 - Requirement’s traceability table

**Entity-relationship diagram**

This is a mongoDB NoSQL database.

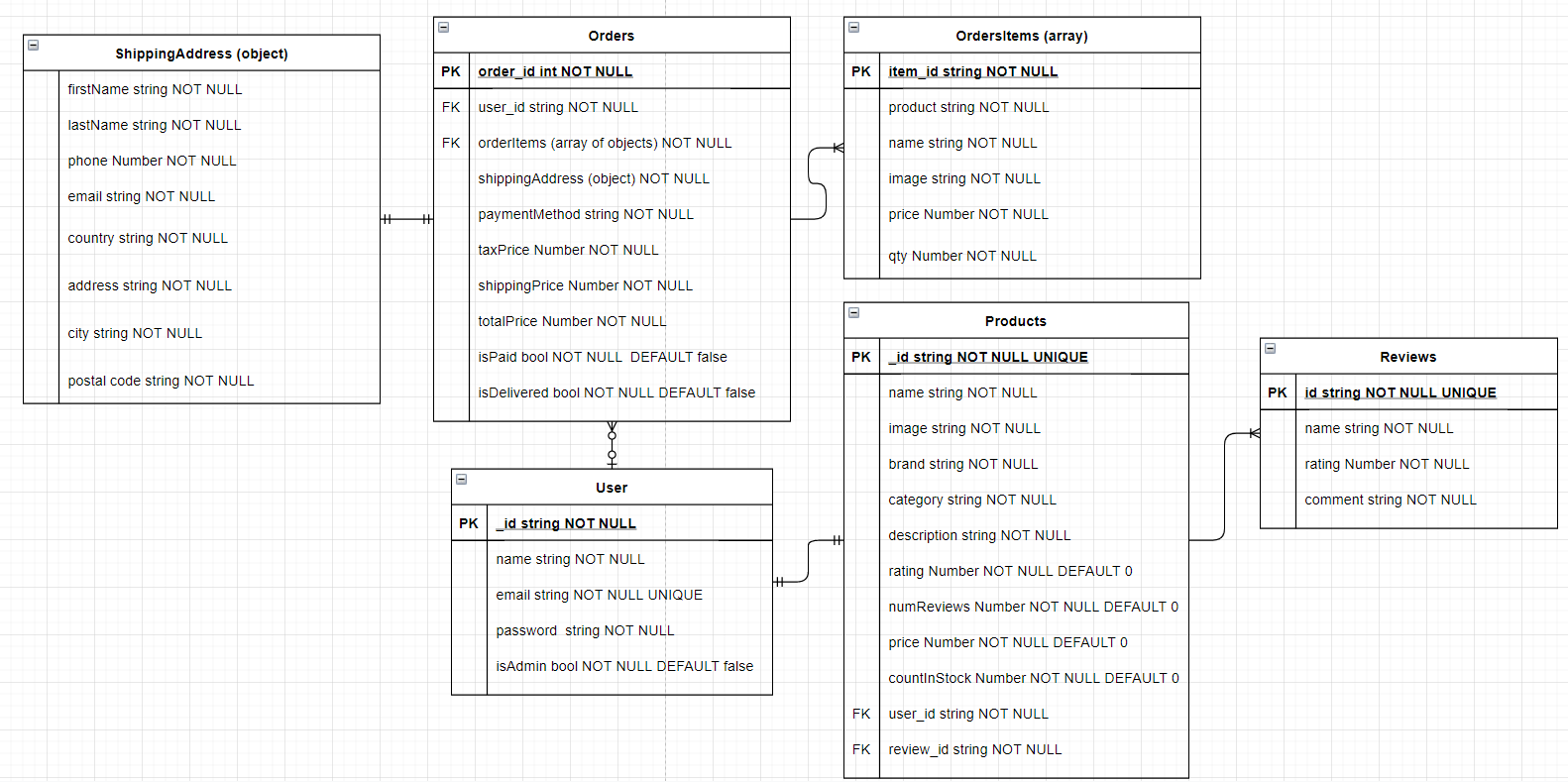


Figure 4 - Entity-relationship diagram

**Structure diagram**

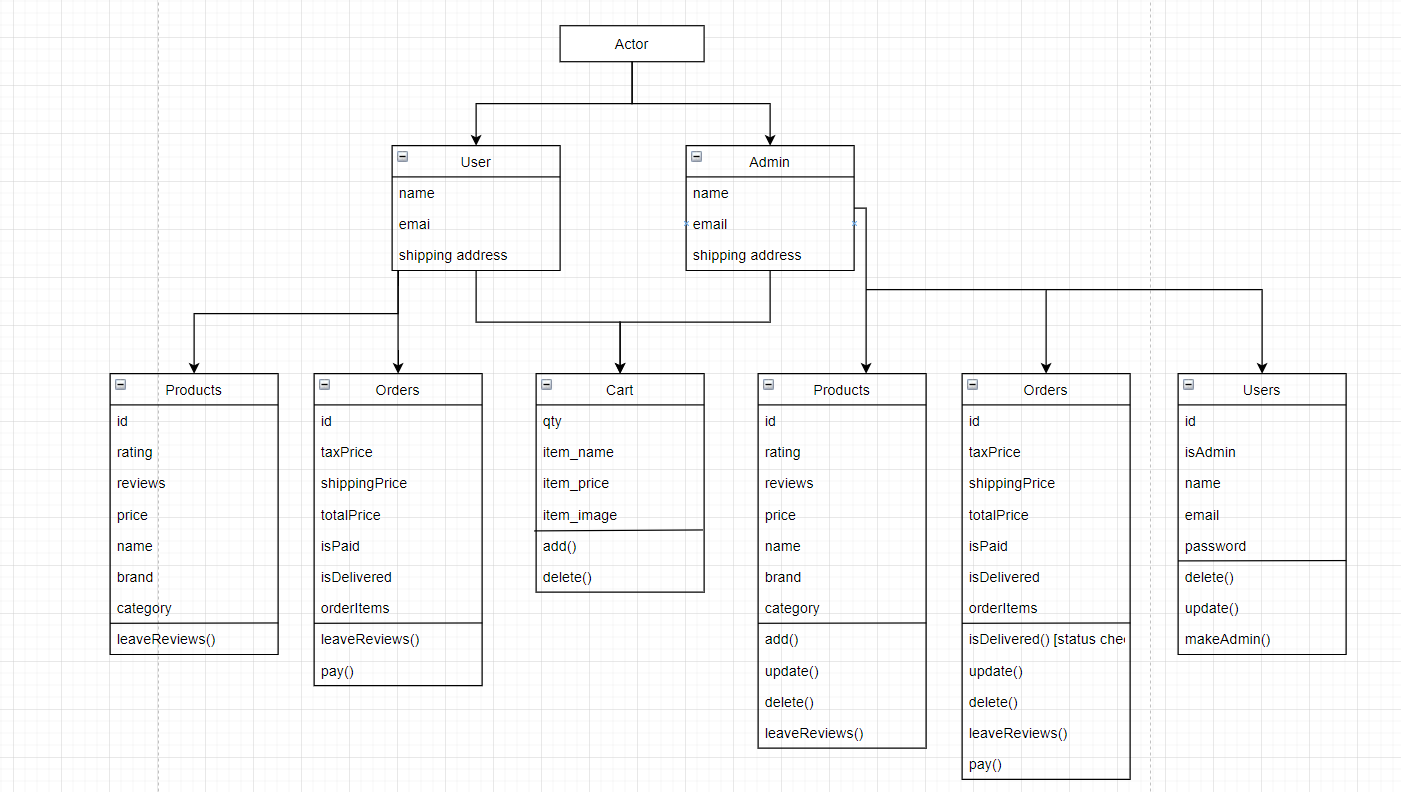


Figure 5 - Structure diagram

**IoT SCANNER IMPLEMENTATION INTO THE PROJECT**

Since we now have our fully working ecommerce shop, there is not so much left to do. We want to implement a scanner that would help us out with our physical part of the store. If the product is bought in store for example, then there is no way to track the quantity in stock left, we would want to scan the product itself and then just write the number we want to subtract from our inventory total number.

For example, one of the customers come to our physical store. We have 50 blue jeans in stock. Our customer wants to buy 3 of them. In that case, we would scan the product with our scanner and input the number we want to subtract from our inventory. In that case when we scanned the product, we would have 47 jeans left in stock.

**How would this work?**

This would be simple, each product in the physical shop would have a QR code with a unique product ID.

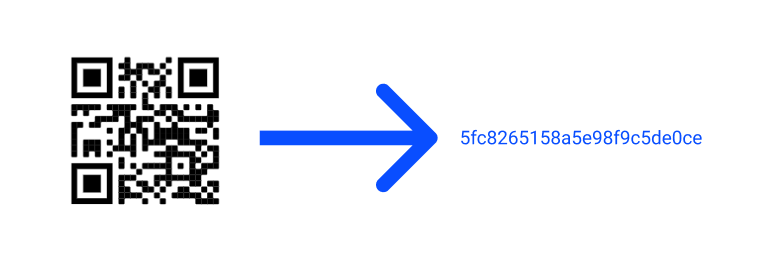


Figure 6 - QR code information

We would scan for this QR code, it would give us the \_id for this product:

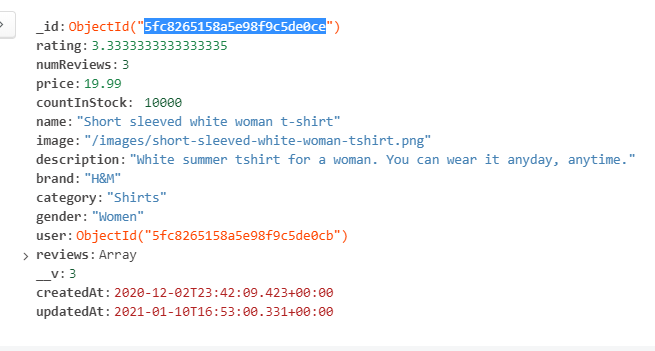


Figure 7 - Object product in the database



Figure 8 - Program screen

And since we have the ID of the product, we can search for the ID in the database and pull the rest of the information that we need out of the database. In this QR scan we would only need the name to confirm that the item is the same and to enter the quantity we want to subtract from the database. This would be the same other way round if a user wants to return the items to the physical shop. All we would have to do is scan and enter the quantity that is being returned.

**SOFTWARE IMPLEMENTATION TOOLS**

Tools that have been used to implement the software:

**React.js + redux** – client side / frontend

**Node.js / Express.js** - server-side / backend.

**MongoDB** – database.

**Jest + Enzyme** – testing.

**VS Code -** code editor for writing code and tests.

**Git + GitHub** – for version control

**USER DOCUMENTATION**

User documentation can be found [here](https://dziugaspeciulevicius.github.io/eds-store-docs/docs/getting-started).

**IMPLEMENTATION**

**Frontend**

**React Setup**

To set up our react app, we want to create an empty folder and open it. We want to open the folder in the terminal. To create a react app we want to run the command `npx create-react-app frontend` [4]

**Git initialize**

Now when we setup our React application we can set up our version control. Now we want to run few commands. [5]

**React-Bootstrap**

To make life easier and just to save time, I used react-bootstrap to make development faster. In this project I used [bootswatch](https://bootswatch.com/). These are basically bootsrap files just customized. Soo, we do not need to install bootstrap, all we do is need react-bootstrap now. [6]

**Header & Footer**

Now that we installed react-bootstrap, we can start writing header component for our app. [7]

**Listing products in our home page**

Now that we have our home page ready, we can start listing and styling our product cards. For now, we are going to use just a javascript file with products array and some photos and later we will setup the database. [8]

**Adding rating to products**

Now we want to edit the `product` component again. First, we want to import Rating component that we will create later. And we want to replace a div with the component. It will take two props: `value` and `text`. [9]

**React router**

We implement react router so we can have different routes, different url's that we could go to in our project. [10]

**Product Page**

Now that we have our Router installed and listed our products in cars, we can now implement a product page where we display the actual product. [11]

**CONCLUSIONS**

In this project I built fully working ecommerce project and explained a way how we could implement a simple inventory management solution for our physical store. There are many tools that can help with everyday inventory management from simple QR scanners to complex guided cars and robots.

**RECOMENDATIONS**

This application is going to be actively developed in the future. There is a lot of room for improvement for IoT integration into the application. As well with a rise of cryptocurrency, it would be wise to implement a way for users to pay for their goods with their coins.

**LITERATURE**

1. Shopify. 2020. Ecommerce Definition - What Is Ecommerce. [online] Available at: <<https://www.shopify.com/encyclopedia/what-is-ecommerce>> [Accessed 6 April 2020].

2. <https://scanco.com/top-five-benefits-of-a-good-inventory-management-strategy/> [Accessed 16th March 2021]

3. Qr-code-generator.com. 2021. *QR Code Generator | Create Your Free QR Codes*. [online] Available at: <<https://www.qr-code-generator.com/>> [Accessed 22 March 2021].

4. React setup. < <https://dziugaspeciulevicius.github.io/eds-store-docs/docs/frontend#react-setup/>>

5. Git initialize. < <https://dziugaspeciulevicius.github.io/eds-store-docs/docs/frontend#git-initialize>/>

6. React-Bootstrap. < <https://dziugaspeciulevicius.github.io/eds-store-docs/docs/frontend#react-bootstrap>/>

7. Header & Footer. < <https://dziugaspeciulevicius.github.io/eds-store-docs/docs/frontend#header--footer>/>

8. Listing products in our home page. < <https://dziugaspeciulevicius.github.io/eds-store-docs/docs/frontend#listing-products-in-our-home-page>/>

9. Adding rating to products. < <https://dziugaspeciulevicius.github.io/eds-store-docs/docs/frontend#adding-ratings-to-products>/>

10. React router. < <https://dziugaspeciulevicius.github.io/eds-store-docs/docs/frontend#react-router>/>

11. Product page. < <https://dziugaspeciulevicius.github.io/eds-store-docs/docs/frontend#product-page>/>