Fontys Hogescholen

Project plan

Online Inventory System

S-CB-ITS3-S3-CB02 - Individual Project

Ömer Faruk GÖKBAK

3782174

Table of Contents

DOCUMENT CHANGE RECORD....2
DESCRIPTION AND PROJECT GOAL....3
USER STORIES....3
WIREFRAMES....5
JUSTIFICATION FOR FRONT-END AND FRAMEWORK SELECTION....7
JUSTIFICATION FOR BACK-END FRAMEWORK AND DATABASE SELECTION....9
-SPRING BOOT....9
-POSTMAN....9
-RELATIONAL DATABASE SELECTION....9
-SONARQUBE....10
ERD DIAGRAM....11
ARCHITECTURE DIAGRAM....11
UML CLASS DIAGRAM....12
CI SETUP DIAGRAM AND CI/CD PIPELINE....13
UX FEEDBACKS AND UPDATES....14

Document Change Record

Version	Comments	Date
0.0.1	Added description and Project Goal	15-09-2020
0.0.2	Added user stories	16-09-2020
0.0.3	Modified the project goal and user stories	19-02-2020
1.0.0	Added Justification for front-end and framework selection	06-10-2020
1.0.1	Added wireframes and extended	06-10-2020
2.0.1	Added Justification for Back-end, Spring-boot Framework	08-11-2020
2.0.2	Added Class Diagram, Justification for API Monitoring software.	10-11-2020
2.0.3	Added Justification for Database selection.	16-11-2020
2.0.4	Added ERD diagram.	20-11-2020
2.0.5	Added SonarQube (quality assurance metrics).	20-11-2020
3.0.1	Added Continuous Integration Setup Diagram.	25-11-2020
3.0.2	Added Architecture diagram.	26-11-2020
3.0.3	Added SonarQube activity diagram.	26-11-2020
4.0.1	Edited current version up to the feedback of mentors.	12-01-2021
4.0.2	Edited UML Class diagram, replaced with first version.	21-01-2021
4.0.3	Added feedbacks, they are already updated.	22-01-2021

Description and Project Goal

Small business, independent brands, founders they all live in an extensive commerce world where small-to-medium operations compete against global ones. Inventory management is a crucial point in this competition. They are widely used in a variety of industries, from manufacturing to utilities, healthcare, education, government, and more.

Inventory management systems make easier and control the process both of flow and maintenance of inventory to ensure that the right amount of inventory is available at the right time and of the right quality. Nevertheless, inventory mismanagement results in cancelled orders due to inventory shortages, lead to revenue losses despite the best efforts of operations managers, employees, and companies.

Online inventory systems are focused on solving mismanagement issues and keeping a log of products. In the third semester, the goal of the individual project is to develop a web application, an online inventory system, in order to help any organization to increase their time efficiency, keep track of their stocks and manage the inventory system.

Within the web application, the products will be stored and then shipped to customers' addresses. All recorded products in the system will have attributes and the exact place where they are stored, whereby the mismanagement will be solved. There will be two types of stakeholders: Admins and the employees. The employees will be responsible for registration, making an existing product entry, and reduce the product when they will be shipped. Admin can check each department separately or the entire stock management and inventory logs.

User Stories

As an Admin;

1- I need an authentication system because I want to give access only my employees.
2- I want to add or delete employees and departments.
3- I need to assign any user to any department if a department needs an extra work(P.S. Administration department only consists of admin).
4- I want to see out of products in homepage, so I can be notified automatically.
5- I want to check inventory history(log) about any specific product to check ups and downs about quantities.
6- I need to check the products shipments by exact date and the address where they were shipped to solve absent product issues. (Could not be implemented due to lack of time.)

As an Employee;

1- I need authentication to access the app I can use the system.
2- I need to edit my personal info(password, email) in order to authentication.

When new delivery arrived:

3- Products have their unique barcode, after checking whether the product exist or not; I need to create a new product in my department if it doesn't exist.
4- If the product exists, I need to update quantities of the existent product to tracking current stocks.
5- I need to create a new pallet to store product inside. Each pallet has unique barcode, name, description and location where it is located.
6- If a pallet is full capacity, then I need to save the pallet with current quantities because if quantity of a specific product in the pallet will be reduced in time, then extra same product can be added. The quantities at the beginning will be saved and kept until the product is removed from the pallet.
7- I need to edit maximum product quantities for each pallet because smaller products can be stored further than the bigger ones. (Could not be implemented due to lack of time.)
8- I want to see out of products inside the pallets in my homepage, so I can be notified automatically and refill the pallet.
9- I need to edit attributes of a product if there will be any changes.

When a product will be shipped to an address:

10- I need to create a new shipping bill, consists of the address, date, barcodes of products and amounts, and every log will be kept in the system so admin can check everything about shipments.(Could not be implemented due to lack of time.)
11- If an entire pallet will be shipped, the shipping bill will consist of the address, date and pallet barcode because the products are already within the pallet. (Could not be implemented due to lack of time.)

Wireframes

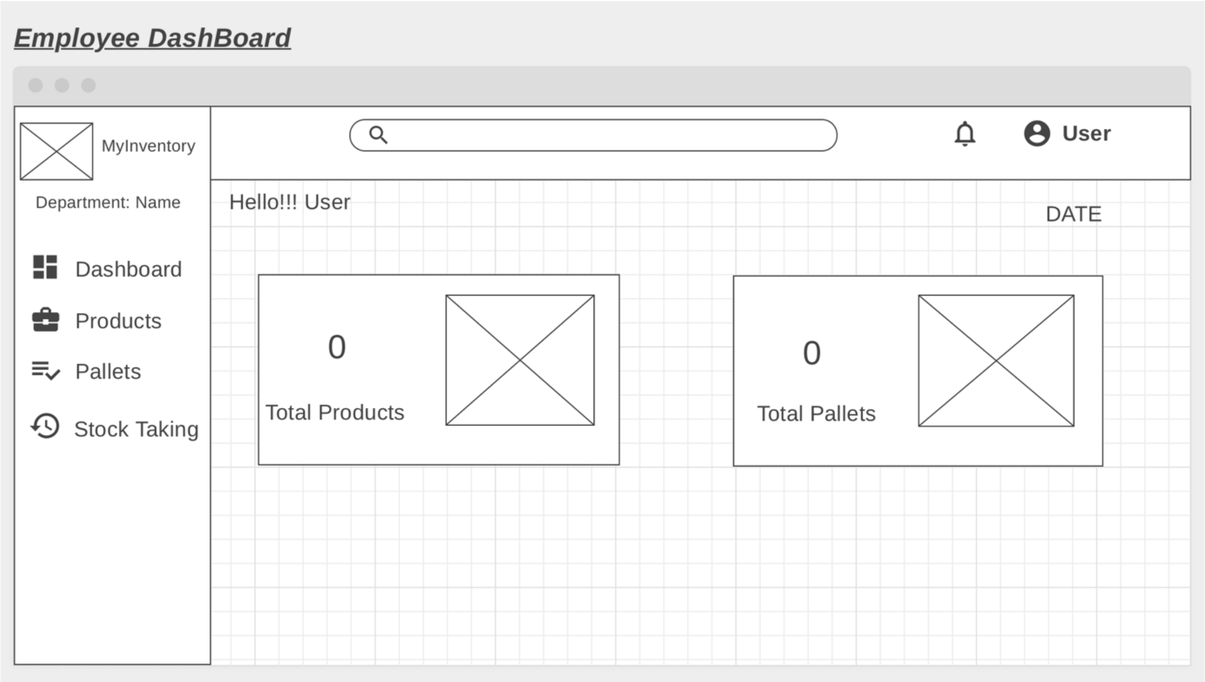


Figure 1

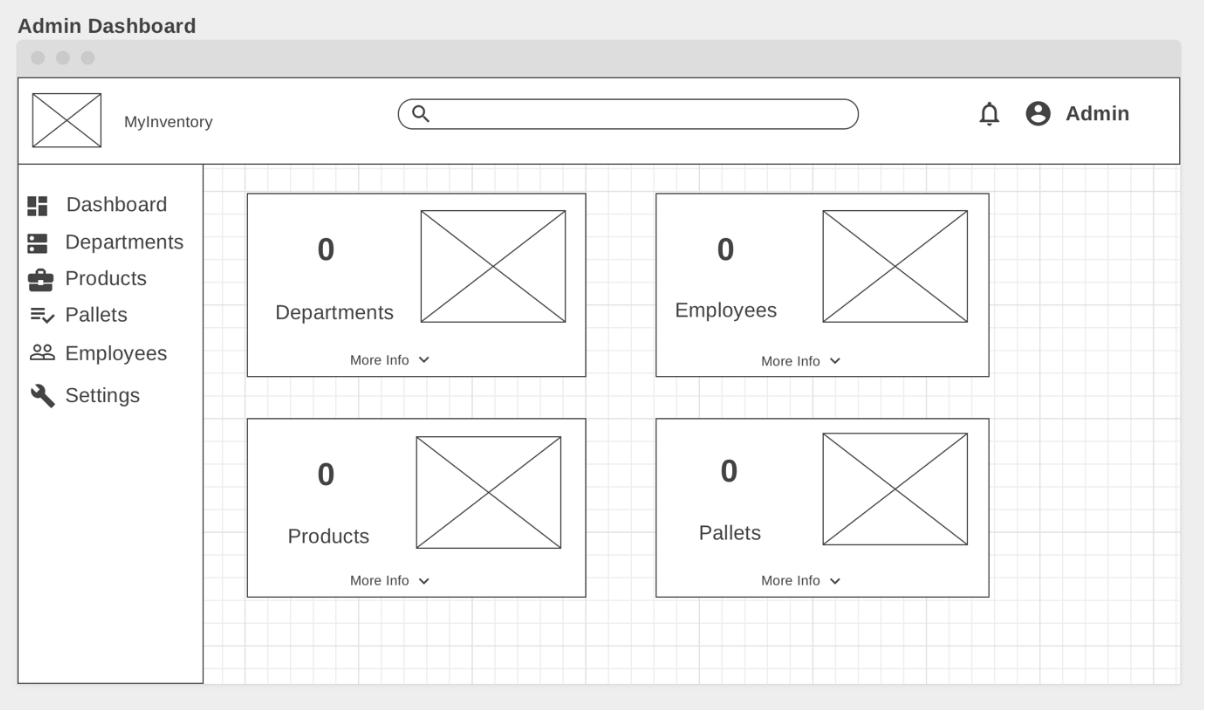
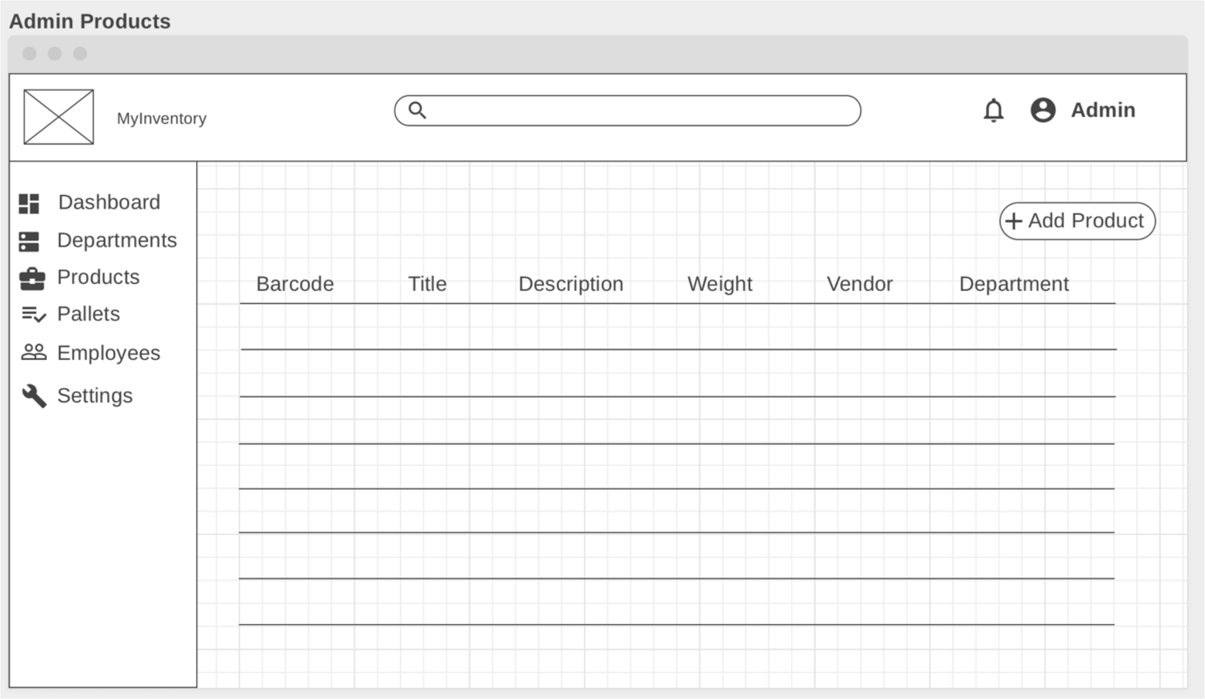
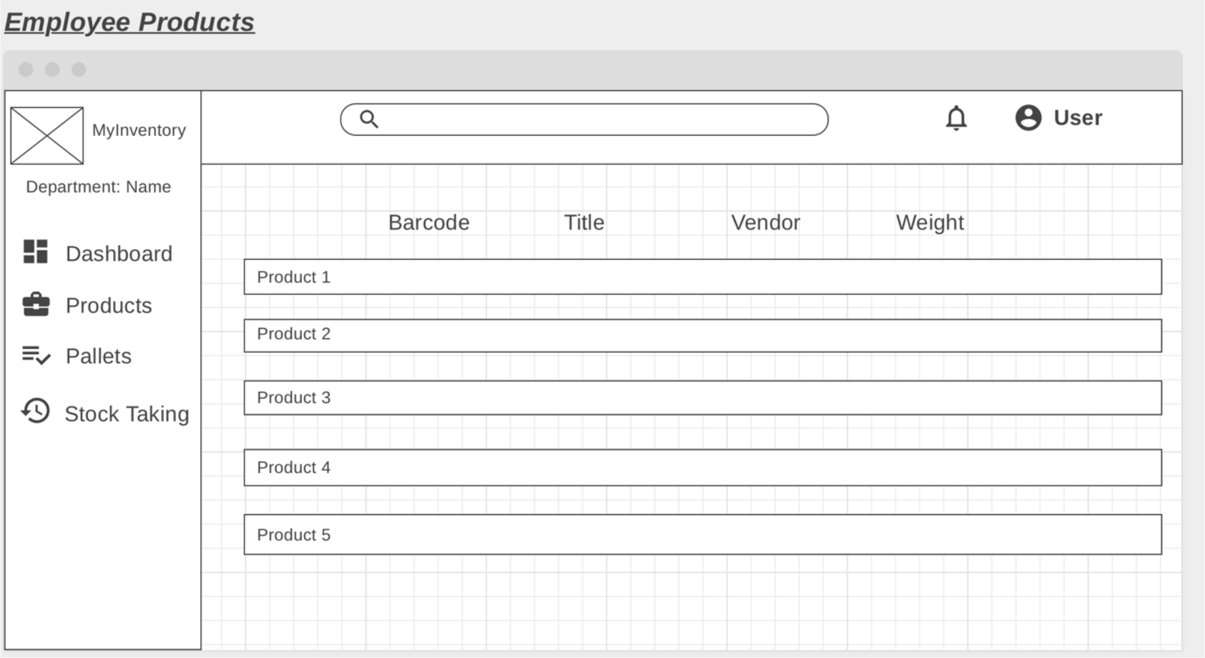


Figure 2



Barcode	Title	Description	Weight	Vendor	Department

Figure 3



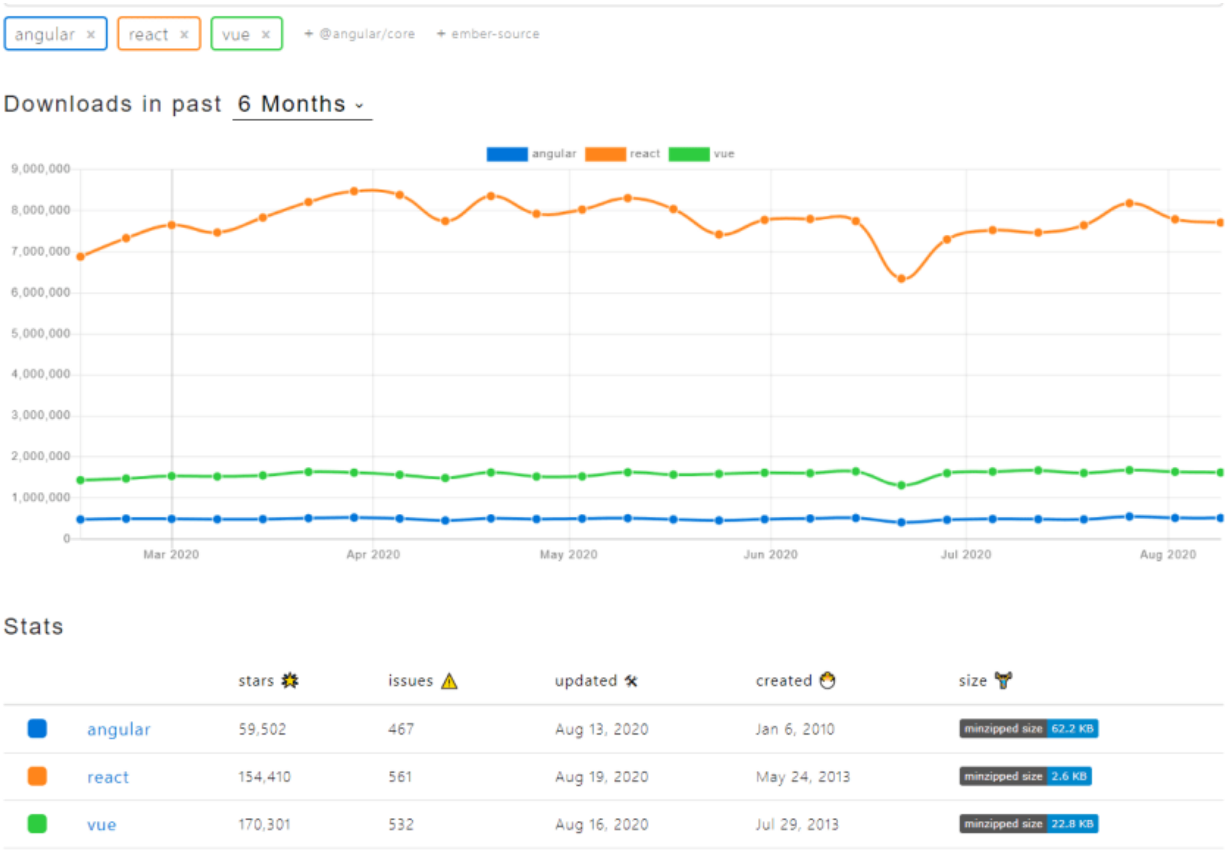
Barcode	Title	Vendor	Weight
Product 1			
Product 2			
Product 3			
Product 4			
Product 5			

Figure 4

Justification for Front-End and Framework Selection

Comparison Angular vs. React vs. Vue

FEATURES	ANGULAR	REACT	VUE
TYPE	A framework	Library to build UI	A library
APLICATION TYPES	Native apps, hybrid apps and web apps	SPA and mobile apps	SPA and native apps
IDEAL FOR	Large-scale, feature-rich apps	Modern web development and-native rendered app for IOS and Android.	Web development and single-page apps.
LEARNING CURVE	A steep curve	A little bit easier than Angular	A small learning curve
DEVELOPER-FRIENDLY	Structure-based framework	Have flexibility in the development environment	Have separation of concerns
MODEL	Based on component based MVC architecture	Based on Virtual DOM(Document Object Model)	Same with React
WRITTEN IN	TypeScript	JavaScript(also TypeScript if preferred)	JavaScript
COMMUNITY SPORT	A Large Community of developers and supporters	Facebook developers' community	Open-source project sponsored through crowdsourcing
LANGUAGE PREFEREFCE	Recommends the use of TypeScript	Recommends the use of JSX – JavaScript XML	HTML templates and JavaScript
UI/DOM MANIPULATING	✓	✓	✓
STATE MANAGEMENT	✓	✓	✓
ROUTING	✓	✗	✓
FORM VALIDATION & HANDLING	✓	✗	✗
HTTP CLIENT	✓	✗	✗



angular × react × vue × + @angular/core + ember-source

Downloads in past 6 Months ⌄

Stats

		stars	issues	updated	created	size
🟦	angular	59,502	467	Aug 13, 2020	Jan 6, 2010	minzipped size 62.2 KB
🟧	react	154,410	561	Aug 19, 2020	May 24, 2013	minzipped size 2.6 KB
🟩	vue	170,301	532	Aug 16, 2020	Jul 29, 2013	minzipped size 22.8 KB

Figure 5(NPM trends Survey: It's about what technology you worked in past, willing to work and willing to learn.)

As a conclusion:

All of them Angular, React.js and Vue are powerful tools for web developers. I tried to create simple application in order to have an opinion on all environments. Overall, I decided to use React for my individual project. Here are some of the reasons:

- There is more JavaScript rather than framework-specific-code.

- The most popular framework for managing application. (see Figure[1])

- The primary language is JavaScript, On the other hand, Angular uses TypeScript and it is unfamiliar to me and it just takes some time to get used to the new one.

- There are a lot of tutorials, resources and support for the framework and I have already started to learn.

- Testing is fast and easy.(Jest)

- Extremely fast,courtesy of React’s Virtual DOM implementation and various rendering optimizations.

- Skills learned in React can be applied to React Native(mobile) development.

Justification for Back-End Framework and Database Selection

-Spring Boot

Spring boot is based on Java, which is one of the world's most popular programming languages and I use it to create my back-end service. Besides that, Spring Boot can help to quickly build any applications without having to worry about safe(spring security) and helps to avoid all the manual work of writing boilerplate code, annotations, and complex XML configurations.It also comes with embedded HTTP servers like Jetty and Tomcat to test web applications. The integration of Spring Boot with the Spring ecosystem which includes Spring Data, Spring Security, Spring ORM, and Spring JDBC is easy. It allows for easily connecting with database and queue services like Oracle, PostgreSQL, MySQL, MongoDB,etc. I use PostgreSQL which makes connection and configuration easier to using Spring. The best option; do not need to struggle with XML based configurations at all. Very much simplified properties. The beans(Component, Repository, Service etc.) are initialized, configured and wired automatically.

-Postman (API Monitoring software)

Postman is a tool that provides a rest client function which is used to share, test, document and monitor APIs without the need for long codes. The reason of why postman did used in this project, was explained in a few steps below.

- It is used for testing the functionality of my REST API.
- No coding required.
- Good and user-friendly UI.
- The variables part is useful.I can easily define the variables.
- With history, I can access any requests easily that have been submitted before.

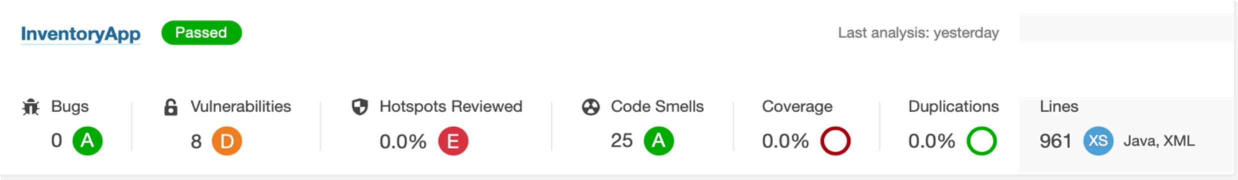
-Relational Database Selection

PostgreSQL is an object-relational database management system, which I need to use and store my relational models in safe. It supports a data model consisting of a collection of named relations.It has been established as a major player in the Open-Source database world and is challenging big players such as Oracle, Sybase, and IBM. PostgreSQL is professionally maintained and developed software, capable of running complex, data-driven applications.

I had only experience with MySQL, therefore I have selected a new one in order to gain experience with different database systems. I have started to work with PostgreSQL GUI tool, pgAdmin4, is easy to use, being free and open source. PgAdmin4 is the most used GUI tool for PostgreSQL and it's the only native PostgreSQL GUI tool as well.

-SONARQUBE

SonarQube is an open-source platform and also a tool that checks whether the code in the project comply with certain rules. While doing this checking it does not compile the code. It is installed as a server and shows the errors in the project. SonarQube also gives examples of how to solve detected issues. Images below, it is declared(Figure 6) that Inventory Application is passed the certain rules and in the next image(Figure 7) process is shown as a chart and with specific dates.

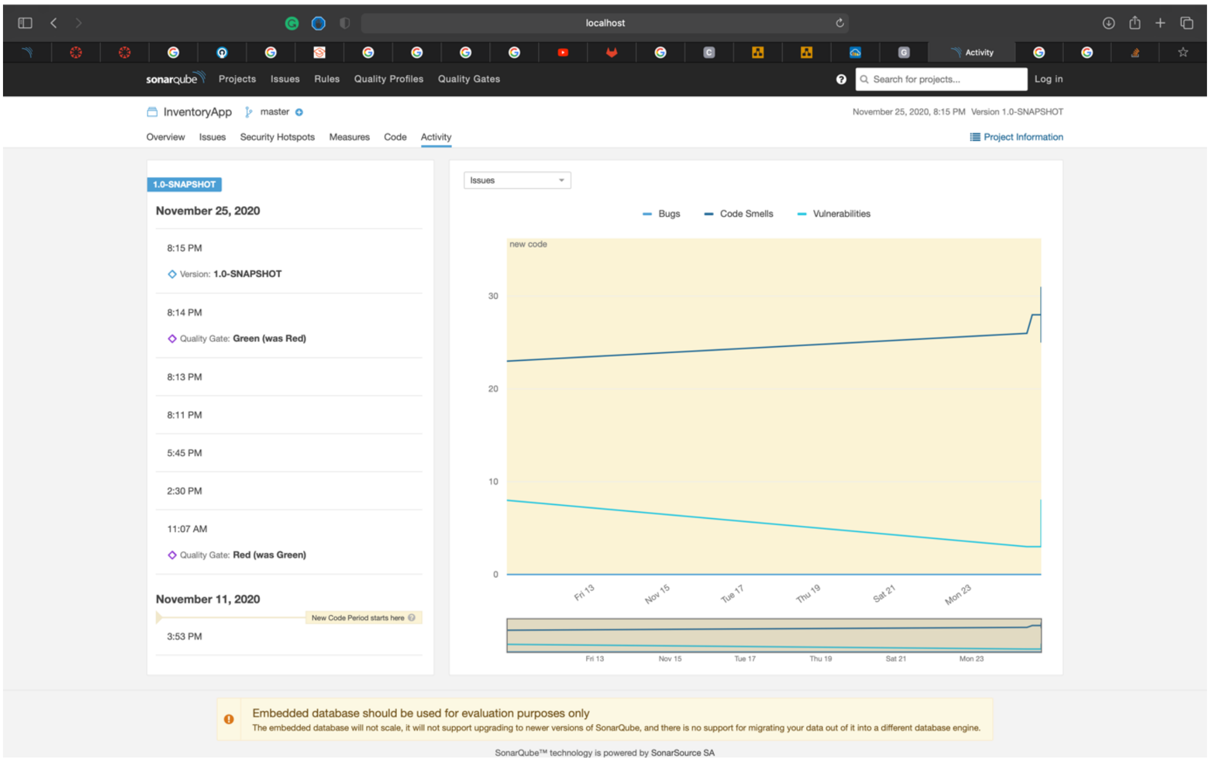


InventoryApp Passed

Last analysis: yesterday

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 A	8 D	0.0% E	25 A	0.0%	0.0%	961 XS Java, XML

Figure 6



November 25, 2020, 8:15 PM Version 1.0-SNAPSHOT

1.0-SNAPSHOT

November 25, 2020

8:15 PM

Version: **1.0-SNAPSHOT**

8:14 PM

Quality Gate: **Green (was Red)**

8:13 PM

8:11 PM

5:45 PM

2:30 PM

11:07 AM

Quality Gate: **Red (was Green)**

November 11, 2020

New Code Period starts here

3:53 PM

Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA

Figure 7

ERD Diagram

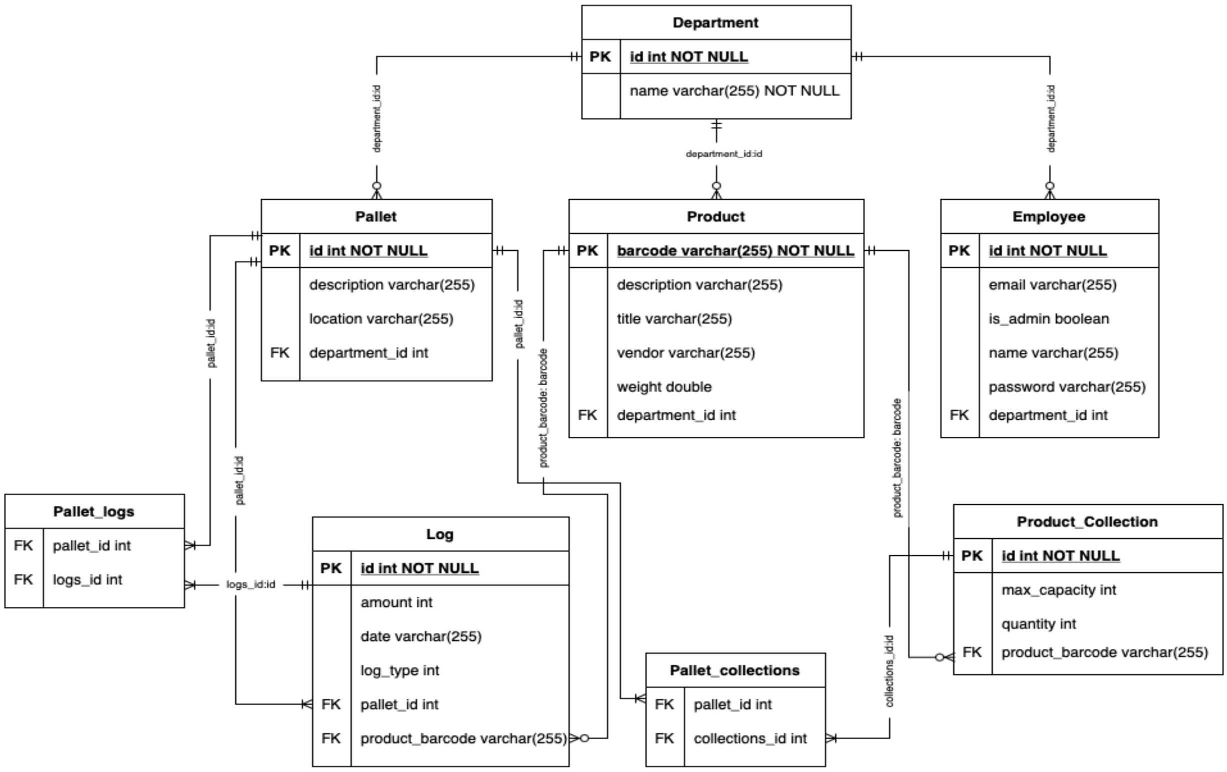


Figure 8

ARCHITECTURE DIAGRAM

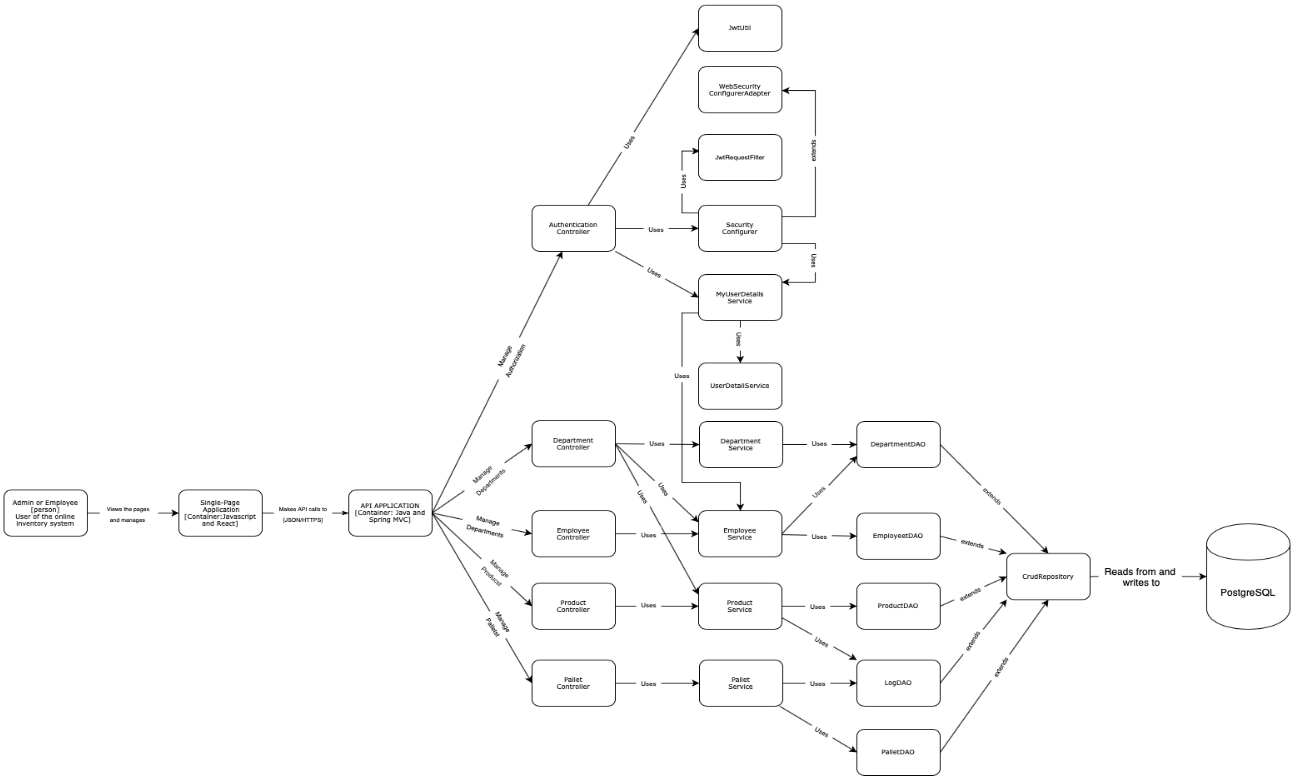


Figure 9

**The diagram above shows the entire process, step-by-step from frontend to backend.*

UML CLASS DIAGRAM

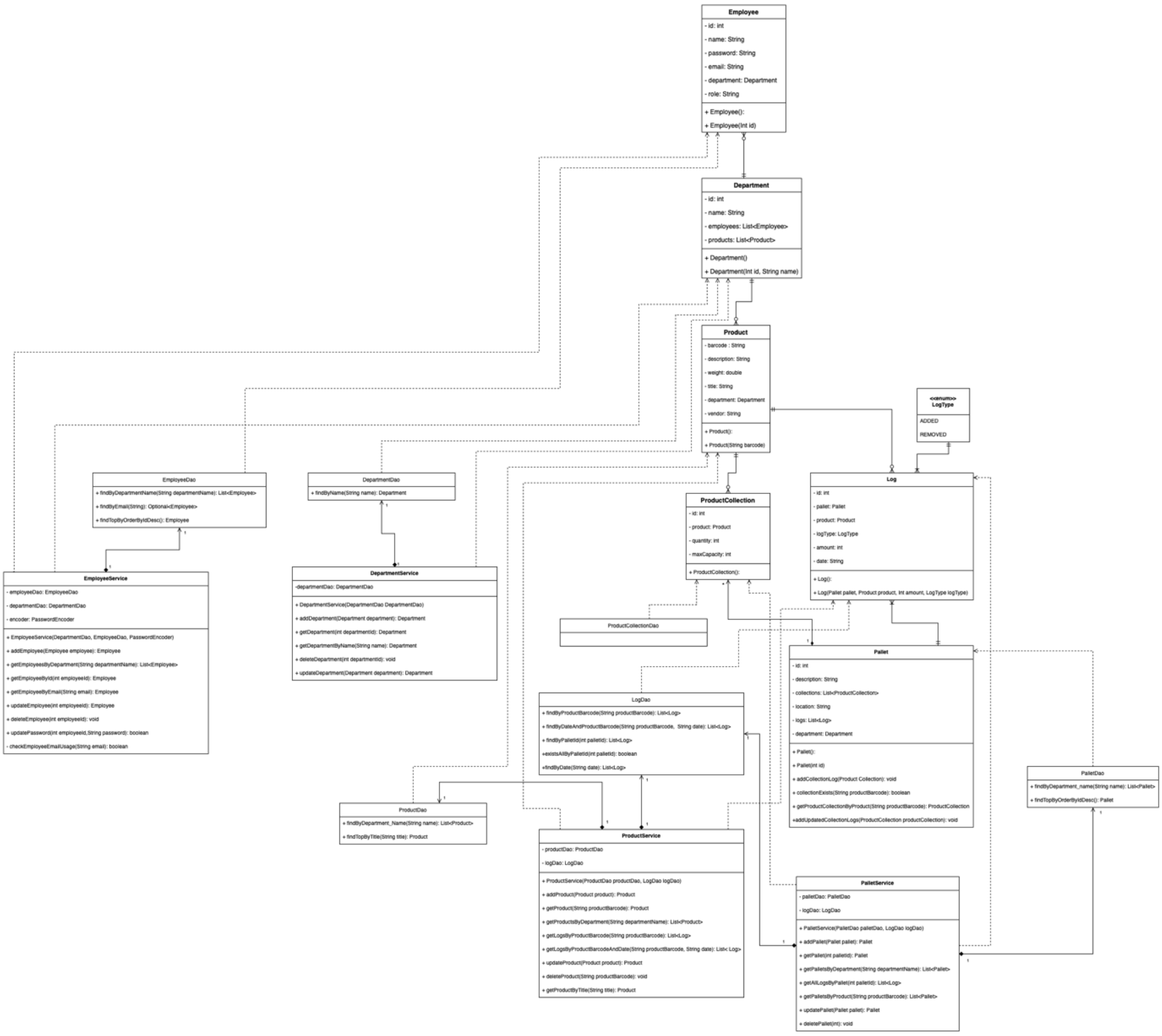

Employee
- id: int
- name: String
- password: String
- email: String
- department: Department
- role: String
+ Employee():
+ Employee(Int id)
Department
- id: int
- name: String
- employees: List<Employee>
- products: List<Product>
+ Department()
+ Department(Int id, String name)
Product
- barcode : String
- description: String
- weight: double
- title: String
- department: Department
- vendor: String
+ Product():
+ Product(String barcode)
<<enum>>
LogType
ADDED
REMOVED
ProductCollection
- id: int
- product: Product
- quantity: int
- maxCapacity: int
+ ProductCollection():
Log
- id: int
- pallet: Pallet
- product: Product
- logType: LogType
- amount: int
- date: String
+ Log():
+ Log(Pallet pallet, Product product, Int amount, LogType logType)
EmployeeDao
+ findByDepartmentName(String departmentName): List<Employee>
+ findByEmail(String): Optional<Employee>
+ findTopByOrderByIdDesc(): Employee
DepartmentDao
+ findByName(String name): Department
EmployeeService
- employeeDao: EmployeeDao
- departmentDao: DepartmentDao
- encoder: PasswordEncoder
+ EmployeeService(DepartmentDao, EmployeeDao, PasswordEncoder)
+ addEmployee(Employee employee): Employee
+ getEmployeesByDepartment(String departmentName): List<Employee>
+ getEmployeeById(int employeeId): Employee
+ getEmployeeByEmail(String email): Employee
+ updateEmployee(int employeeId): Employee
+ deleteEmployee(int employeeId): void
+ updatePassword(int employeeId,String password): boolean
- checkEmployeeEmailUsage(String email): boolean
DepartmentService
-departmentDao: DepartmentDao
+ DepartmentService(DepartmentDao DepartmentDao)
+ addDepartment(Department department): Department
+ getDepartment(int departmentId): Department
+ getDepartmentByName(String name): Department
+ deleteDepartment(int departmentId): void
+ updateDepartment(Department department): Department
ProductCollectionDao
Pallet
- id: int
- description: String
- collections: List<ProductCollection>
- location: String
- logs: List<Log>
- department: Department
+ Pallet():
+ Pallet(int id)
+ addCollectionLog(Product Collection): void
+ collectionExists(String productBarcode): boolean
+ getProductCollectionByProduct(String productBarcode): ProductCollection
+addUpdatedCollectionLogs(ProductCollection productCollection(: void
LogDao
+ findByProductBarcode(String productBarcode): List<Log>
+ findByDateAndProductBarcode(String productBarcode, String date): List<Log>
+ findByPalletId(int palletId): List<Log>
+existsAllByPalletId(int palletId): boolean
+findByDate(String date): List<Log>
PalletDao
+ findByDepartment_name(String name): List<Pallet>
+ findTopByOrderByIdDesc(): Pallet
ProductDao
+ findByDepartment_Name(String name): List<Product>
+ findTopByTitle(String title): Product
ProductService
- productDao: ProductDao
- logDao: LogDao
+ ProductService(ProductDao productDao, LogDao logDao)
+ addProduct(Product product): Product
+ getProduct(String productBarcode): Product
+ getProductsByDepartment(String departmentName): List<Product>
+ getLogsByProductBarcode(String productBarcode): List<Log>
+ getLogsByProductBarcodeAndDate(String productBarcode, String date): List< Log>
+ updateProduct(Product product): Product
+ deleteProduct(String productBarcode): void
+ getProductByTitle(String title): Product
PalletService
- palletDao: PalletDao
- logDao: LogDao
+ PalletService(PalletDao palletDao, LogDao logDao)
+ addPallet(Pallet pallet): Pallet
+ getPallet(int palletId): Pallet
+ getPalletsByDepartment(String departmentName): List<Pallet>
+ getAllLogsByPallet(int palletId): List<Log>
+ getPalletsByProduct(String productBarcode): List<Pallet>
+ updatePallet(Pallet pallet): Pallet
+ deletePallet(int): void

Figure 10

**UML diagram(Figure 10) is also uploaded to the documentation repository to see better.*

CI Setup diagram and Ci/Cd Pipeline

The diagram(Figure 11) below indicates the steps in the GitLab continuous integration process, and the picture(Figure 12) represents how a single passed pipeline looks in GitLab Ci/Cd pipelines.

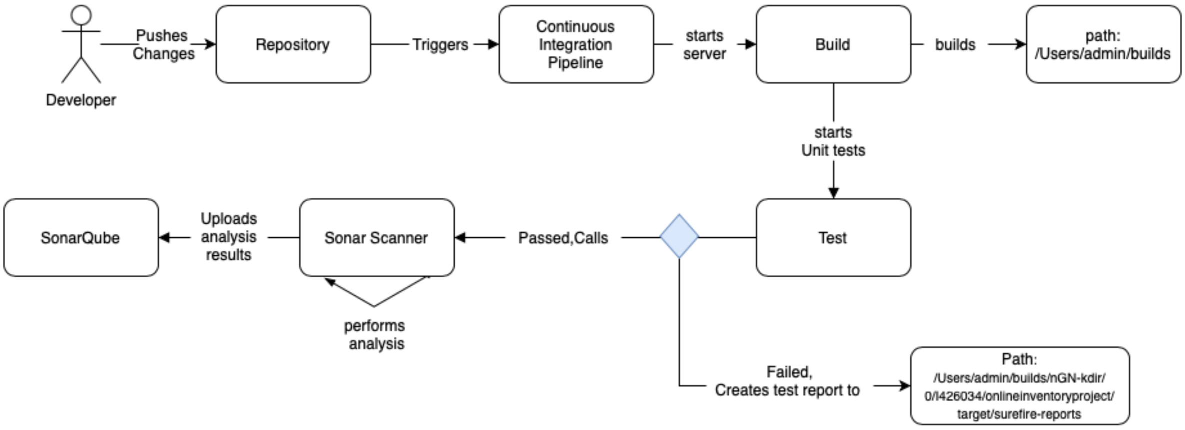
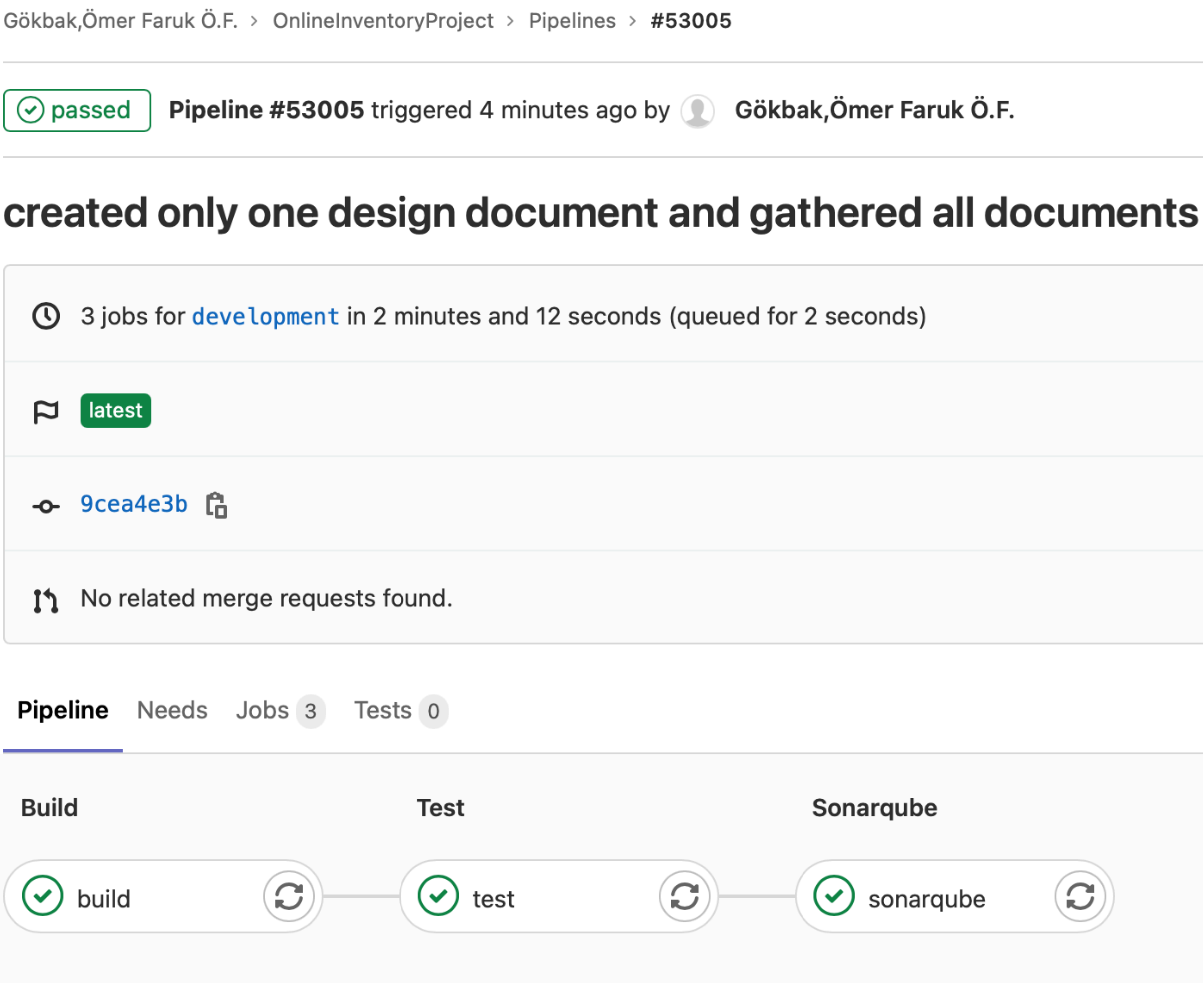

Developer
Pushes Changes
Repository
Triggers
Continuous Integration Pipeline
starts server
Build
builds
path: /Users/admin/builds
starts Unit tests
Test
Passed,Calls
Sonar Scanner
performs analysis
Uploads analysis results
SonarQube
Failed, Creates test report to
Path: /Users/admin/builds/nGN-kdir/0/l426034/onlineinventoryproject/target/surefire-reports

Figure 11



Gökbak,Ömer Faruk Ö.F. › OnlineInventoryProject › Pipelines › **#53005**

✓ passed **Pipeline #53005** triggered 4 minutes ago by **Gökbak,Ömer Faruk Ö.F.**

created only one design document and gathered all documents

3 jobs for `development` in 2 minutes and 12 seconds (queued for 2 seconds)

latest

`9cea4e3b`

No related merge requests found.

Figure 12

UX Feedbacks and Updates

Below are feedbacks from two different peer from the class. Application is updated on the feedbacks given. Async requests are used in React application. I have implemented web sockets for my REST – API but did not use it on Client side because lack of time and I don't think it is a requirement for my application.

Peer 1 – Hristov Sava

1. Add remove/delete buttons near to each data to the pages where lists are displayed.
2. Add and edit button to Settings page same as in Product page to prevent update personal info mistakenly.
3. When you inform the user as password is updated , use the green color for text to show that it is completed successfully.
4. Check the spelling and rename the logout button.
5. When you clicked to any card in the dashboard page it should be redirected to the certain page.
6. Remove the "List" from each title of pages in use. Do not use programming words on your screen for the users.

Overall;

- Application is crated with a user-friendly GUI.
- Straight forward and no excessive stuff.
- Great colors for the Navbar-Sidebar and Dashboard design.
- Edit button in the Product Page prevents mistakenly moves.

Peer 2 – Nick Goedhart

1. In the employees page, employee id selection should link to the employee page.
2. In the employees page, department selection should link to the department' employees page.
3. In the product page, department selection should link to the department' products page.
4. In the product page, vendor selection should display the products have the same vendor.
5. In the product page, title selection should also link to the product page same as barcode.
6. In the settings page, editing should be enabled after clicking a button same as the product page to prevent the mistakenly actions.
7. In the products page, products should be removed by the button clicking.

Overall;

- It is a clean website.
- Injections by URL are prevented, and it is really nice touch.
- Application is simple to use.
- Usage of cards or the sidebar menu, interaction simple and easy to use(self-explanatory).