Design Document

1. *Introduction*

1.1. Purpose of the document

The software design document is meant to provide a clear description of the web application for the individual project of the 3rd semester. The document's purpose is to provide a clear understanding of what needs to be built and how it is expected to look at the end.

The design document will cover the frontend as well as the backend of the software, thus giving a clear insight into every aspect of the design of the application.

1.2. Document Overview

This document serves as a description of the functionality and software design for the" ComfyShop" application.

2. *System Overview*

For the 3rd semester of ICT & Software Engineering, students were asked to make a full-stack web application. The project must contain certain guidelines and requirements by using the agile software development methodology. The web application will work as an e-commerce platform that enables customers to browse, search, and purchase products from different categories online. Customers will be able to add products to their cart and view their order history. Additionally, the application will provide the company with access to manage product listings, and update prices and descriptions.

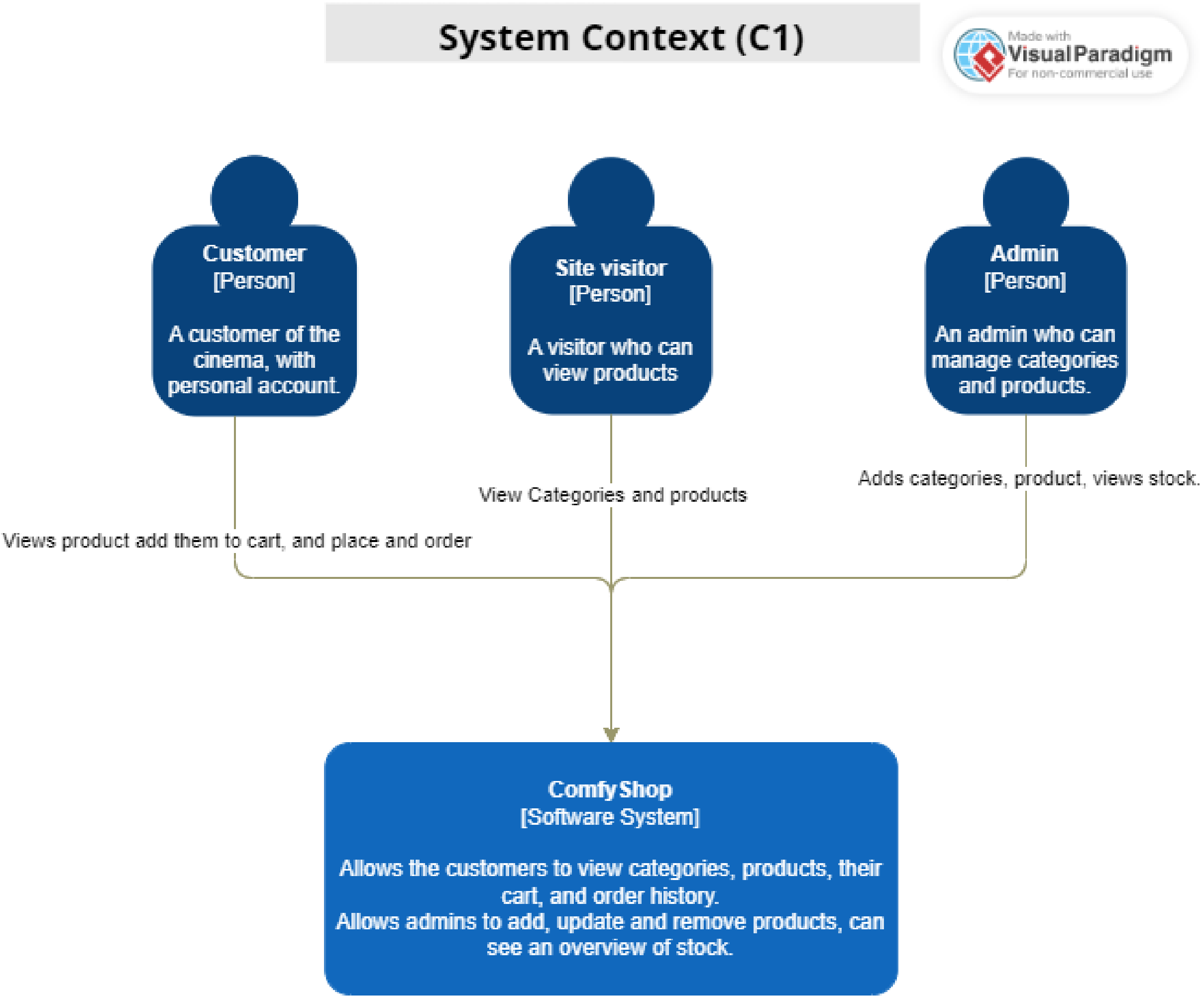
3. System Architecture

3.1. Architectural Design

The application is developed keeping in mind the SOLID principles and ensuring that security is provided to the users. Interfaces are used to ensure the connection to the database in order to avoid repetition in the implementation. The back-end of the application is connected with the front-end by using HTTP requests with AXIOS in order to connect to the REST endpoints. The C4 diagram below shows how the system works

The architecture design consists of 4 levels: System Context(C1), Container (C2), Components (C3), and Code (C4).

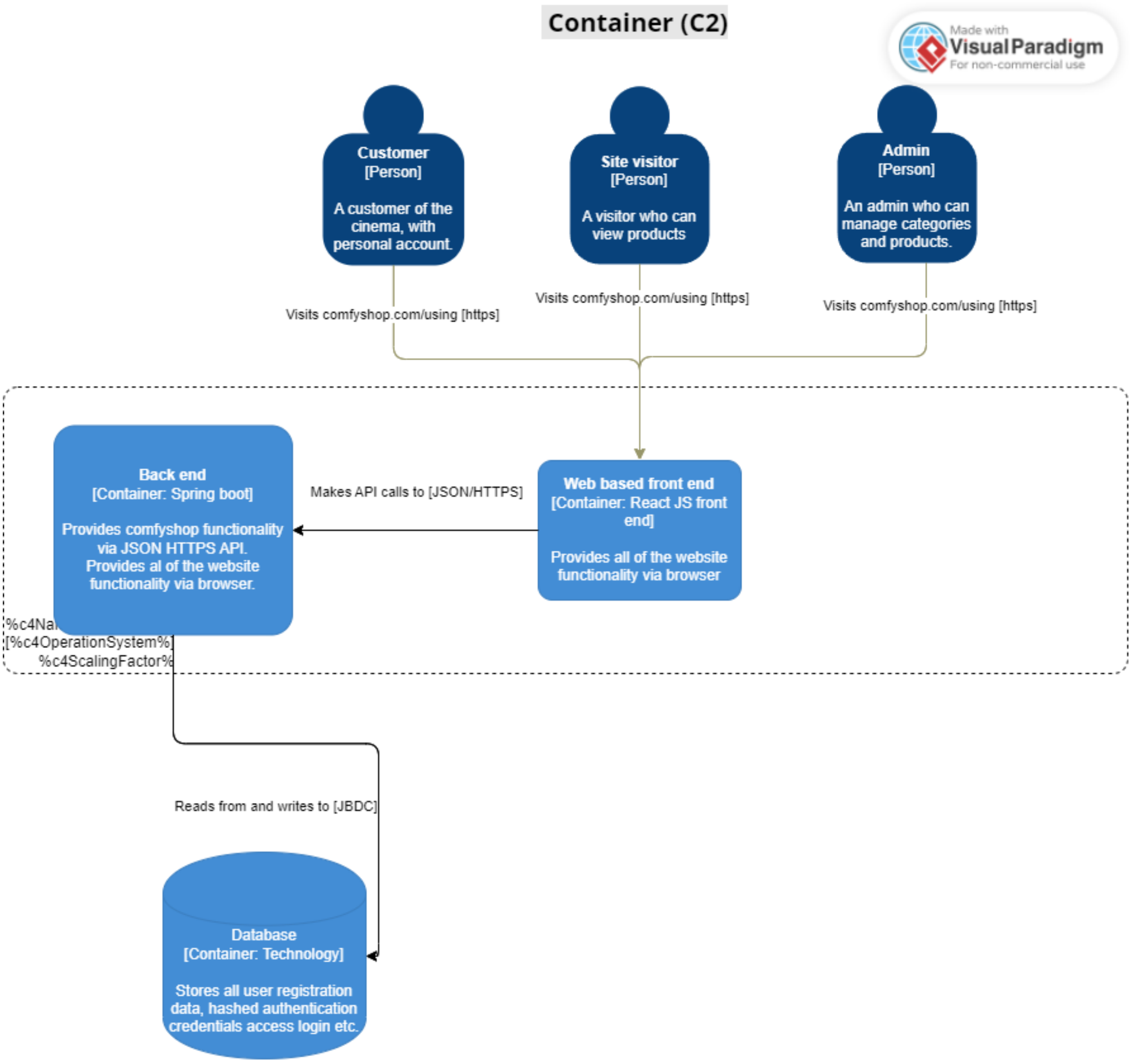
Level 1: System Content


System Context (C1)
Made with Visual Paradigm For non-commercial use
Customer [Person] A customer of the cinema, with personal account.
Site visitor [Person] A visitor who can view products
Admin [Person] An admin who can manage categories and products.
Views product add them to cart, and place and order
View Categories and products
Adds categories, product, views stock.
Comfy Shop [Software System] Allows the customers to view categories, products, their cart, and order history. Allows admins to add, update and remove products, can see an overview of stock.

The above diagram shows the various types of users of the application, namely the Site Visitor, Client, and Administrator, each user has a distinct type of accessing the website.

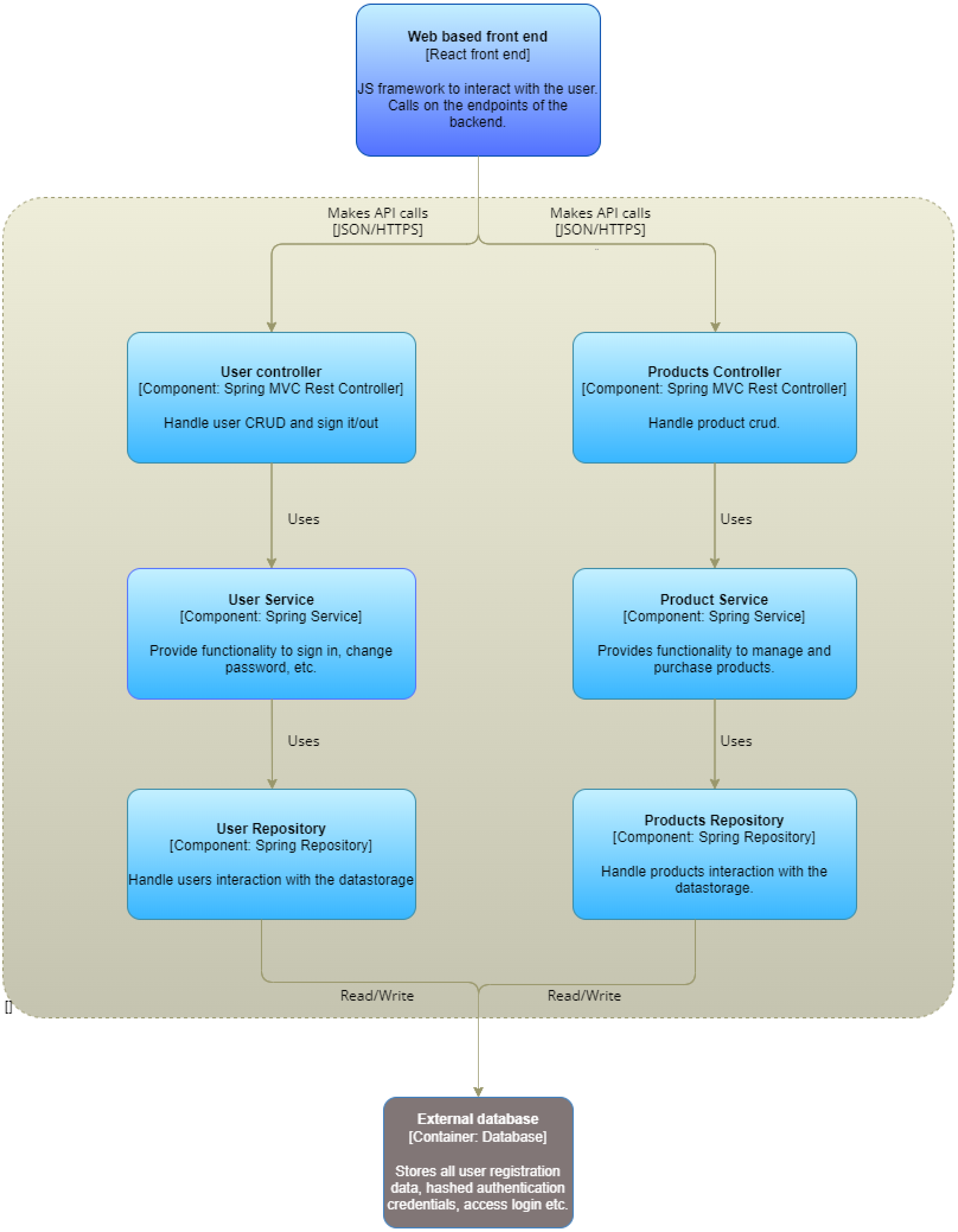
The ComfyShop System is the main software part that enables the client to browse various categories and products, add them to their cart, and place an order. The administrator is responsible for managing the categories, products, and other related tasks.

Level 2: Containers

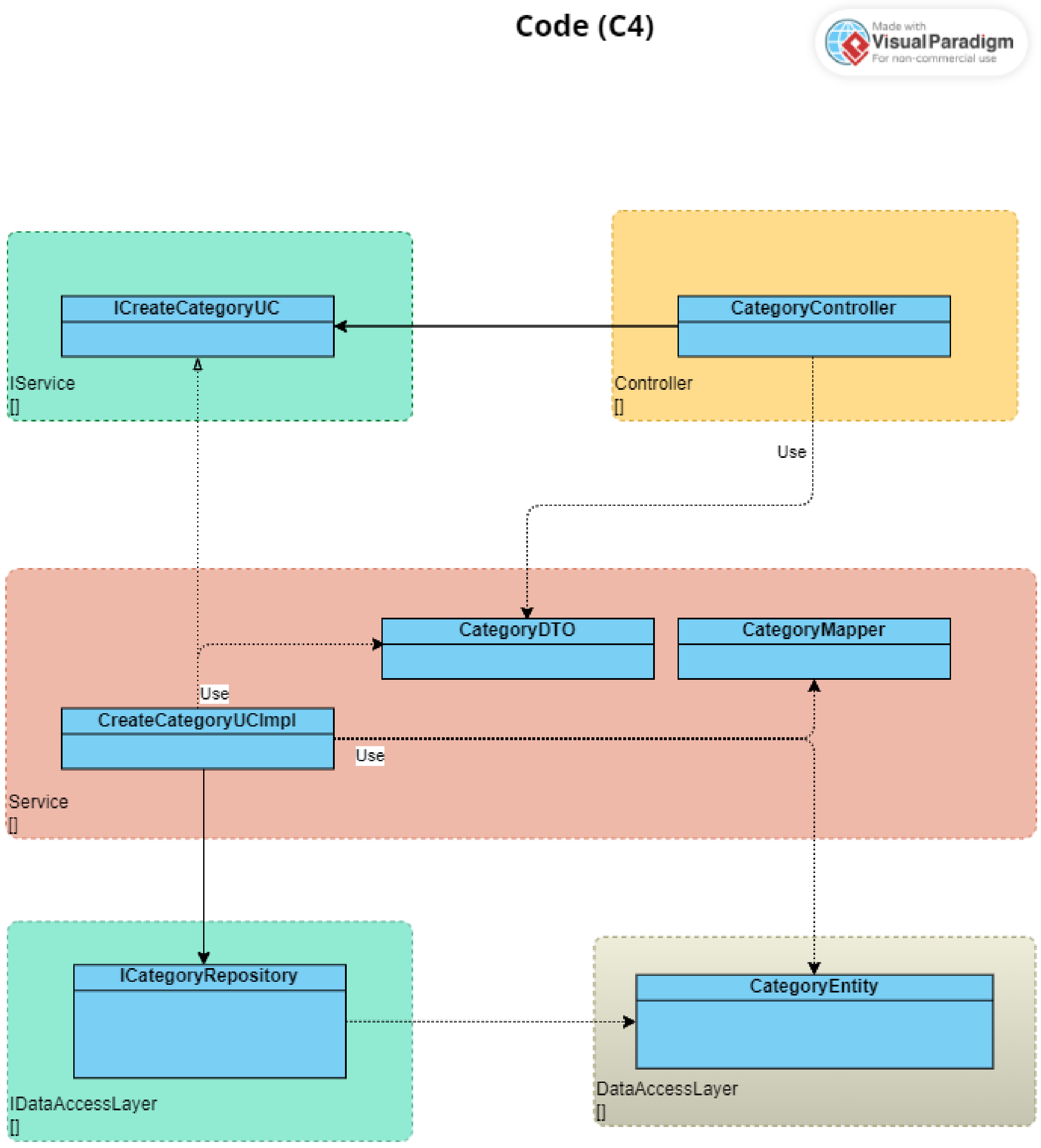

Container (C2)
Made with Visual Paradigm For non-commercial use
Customer
[Person]
A customer of the cinema, with personal account.
Site visitor
[Person]
A visitor who can view products
Admin
[Person]
An admin who can manage categories and products.
Visits comfyshop.com/using [https]
Visits comfyshop.com/using [https]
Visits comfyshop.com/using [https]
Back end
[Container: Spring boot]
Provides comfyshop functionality via JSON HTTPS API.
Provides al of the website functionality via browser.
Makes API calls to [JSON/HTTPS]
Web based front end
[Container: React JS front end]
Provides all of the website functionality via browser
%c4Na
[%c4OperationSystem%]
%c4ScalingFactor%
Reads from and writes to [JBDC]
Database
[Container: Technology]
Stores all user registration data, hashed authentication credentials access login etc.

The diagram describes the relationship between different users and systems, as well as the connection between the back-end and front-end with the database. The front-end container is responsible for providing the user interface (UI) and all website functionalities within the browser. Users and administrators use the UI to interact with the back-end, which provides all the data and logic for the website using Spring Boot and JSON. The back-end makes the front-end functionalities possible by processing the user requests and providing responses accordingly. Finally, the database stores all the data related to users/content and directly interacts with the back-end to ensure the availability of the required data.

Component (C3)


Web based front end
[React front end]
JS framework to interact with the user. Calls on the endpoints of the backend.
Makes API calls
[JSON/HTTPS]
Makes API calls
[JSON/HTTPS]
User controller
[Component: Spring MVC Rest Controller]
Handle user CRUD and sign it/out
Products Controller
[Component: Spring MVC Rest Controller]
Handle product crud.
Uses
Uses
User Service
[Component: Spring Service]
Provide functionality to sign in, change password, etc.
Product Service
[Component: Spring Service]
Provides functionality to manage and purchase products.
Uses
Uses
User Repository
[Component: Spring Repository]
Handle users interaction with the datastorage
Products Repository
[Component: Spring Repository]
Handle products interaction with the datastorage.
Read/Write
Read/Write
[]
External database
[Container: Database]
Stores all user registration data, hashed authentication credentials, access login etc.

Level 3 represents components that are used in the functionality of the application. Controllers are used in order to connect the back end with front-end. With the help of interfaces, the connection with the service components is flexible. Service components are the logic that the controllers basically use. They are connected to data access layers and can CRUD data. The database is used to store all the data of all users/content and is interacting with the backend.



4. *Software Choices*

- ### Spring Boot

Spring Boot is a good framework to get started with the back-end part for full-stack web development, especially for beginners thanks to the fact that it reduces the development process and increases efficiency by having a default setup for unit and integration tests. Therefore, thanks to its features, we are obliged to use Spring Boot this semester for this individual project.

Spring Boot offers some advantages such as productivity, and reduction of development time and it's easy to understand.

The most important thing about Spring Boot would be that everything is auto-configured, so there is no need for manual configurations.

- ### Separating Concerns

Separating concerns stands for splitting a computer program into distinct sections. That would mean that each component of the app has to follow a layered design (e. g. 3 presentation layer – logic layer – data access layer). By using this design principle, the code is easier to understand and can be extended with new features.

- ### React JS

Nowadays, there are a lot of frameworks to choose from, ranging from React JS, Angular, Vue JS, Django etc. each one having its unique functionalities depending on the complexity of the project. Since, ComfyShop is not a very complex project, it has been decided to use React JS. This framework works with components, each function being considered as a component. Therefore, separating concerns is crucial. For beginners, React is much more easier to get started with comparing to the other mentioned frameworks, this being the reason why we have been asked to use it for this semester by our teachers. Also, React is quick, efficient, and makes creating front-end more easy. Thanks to the fact it has a large community, you can easily find anything about different problems you might encounter while creating your application.

- ### MySQL

MySQL is a popular and widely used relational database management system. It is reliable and easy to use.

Moreover, MySQL is free and open-source, everyone can use it for small business.

- Usage of JPA

As mentioned in the software choices, Spring Boot will be used for the backend of this application. In order to retrieve the data from the database, it has been decided to use JPA instead of the classical JDBC so that the needed effort for the data access layers would be reduced significantly. It provides simple queries (for example, if I want to retrieve a user by id, I will just make a method called findUserById and JPA will run the query automatically). It also allows the possibility to create custom queries to retrieve data.

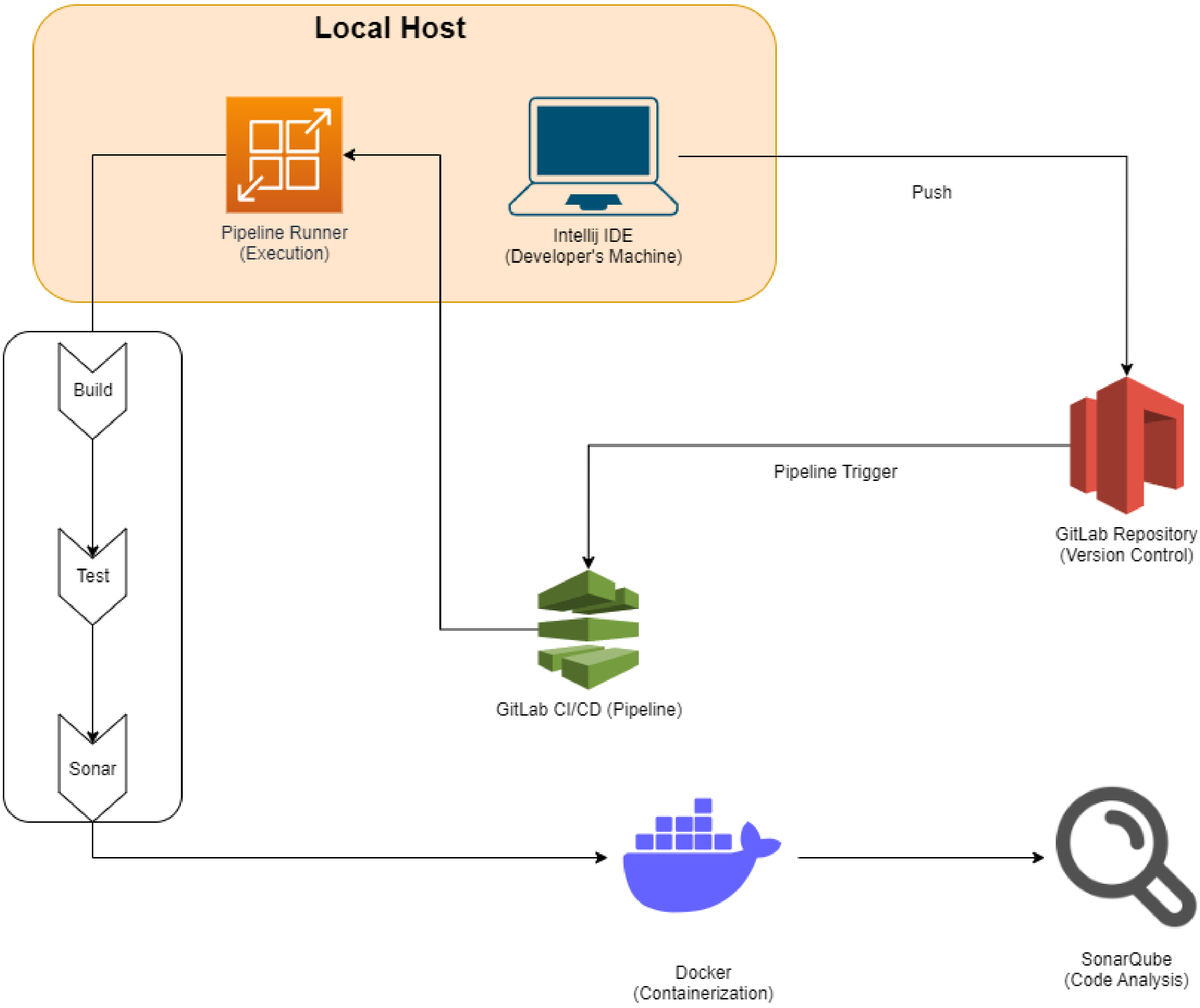
- WebSocket

In order to make the application more interactive, it has been decided to use WebSocket to create a notification system so that users can be aware when a new product has been added and might purchase it. The popup will display on the right/left side of the home page.

- Storing pictures for products

In order to make the application more interactive, it has been decided to use "Cloudinary" which is utilized as a cloud-based storage service to handle the upload and storage of pictures for products in the application. It provides a convenient way to store, manage, and serve images, allowing for efficient storage and retrieval of pictures associated with products.

- CI/CD Pipeline Diagram



I use IntelliJ, a reliable coding tool, to write and edit code for the application. Before saving my changes to Git, I use IDEs to write and test the code, making sure it works correctly.

Git is a central hub for version control, tracking my changes and facilitating collaboration and code sharing.

GitLab, the CI/CD server I use, manages continuous integration and deployment. It monitors my Git repository for new commits and initiates the CI pipeline when changes are detected, ensuring synchronization.

The pipeline runner, part of the CI/CD server, executes the CI pipeline. It retrieves the latest code from Git, and handles tasks like compiling Java code, and running tests.

The pipeline runner creates a specific environment for building and testing the application. This environment includes all the necessary dependencies, libraries, and tools needed to build and run the Java application.

During the build phase, the pipeline runner compiles the Java source code, resolves dependencies, and produces executable binaries.

After the build, the pipeline runner performs automated tests for the application, including both unit tests and integration tests.

I use SonarQube, a static code analysis tool, to check the quality of the codebase. It can be integrated into the CI pipeline to analyse the Java code for issues such as code smells, bugs, vulnerabilities, and more.