|  |
| --- |
| **Fontys - University of Applied Sciences - ICT** |
| Individual track project: Online inventory system |
| Individual track Design Document |

|  |
| --- |
| Martin Georgiev  Student number: 3782484  Eindhoven, Friday, October 9, 2020 |

Document Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| *Date* | *Version* | *Author* | *Comments* |
| 09-10-2020 | 1.0 | **Martin Georgiev** | Added the initial body structure and text of the design document |
|  |  |  |  |

Table of contents

[1 Introduction 4](#_Toc53163625)

[1.1 Document Purpose 4](#_Toc53163626)

[1.2 Document Overview 4](#_Toc53163627)

[2 Choice of front-end language & framework 5](#_Toc53163628)

[2.1 Why React? 5](#_Toc53163629)

[2.1.1 Time and cost-efficiency 5](#_Toc53163630)

[2.1.2 Reusable components 6](#_Toc53163631)

[2.2 The disadvantages of React 6](#_Toc53163632)

[3 Choice of back-end language & framework 7](#_Toc53163633)

[4 Architecture Models 8](#_Toc53163634)

[4.1 System Context (C1) 8](#_Toc53163635)

[4.2 Containers and Technology (C2) 9](#_Toc53163636)

[4.3 Components (C3) 10](#_Toc53163637)

[4.4 Code (C4) 11](#_Toc53163638)

[5 Project UX Design 12](#_Toc53163639)

# Introduction

## Document Purpose

The purpose of this document is to provide an in-depth look at the project’s design structure and choices, part of the individual track. The aim is to provide a concise and well-structured document with more information on the project’s choice of front-end and back-end languages and frameworks, the architecture models that define the project’s scope and a simple UX design that should provide a clearer picture of the final product.

## Document Overview

This document will go through all sections, considered essential to all partaking stakeholders (in this case – teacher/mentor and student). Namely, these sections are:

* **Choice of front-end language/framework** – More information on my choice for a front-end language and framework/library with justification as to why I believe that this is the best choice for this project.
* **Choice of back-end language/framework** – More information on my choice for a back-end language and the framework/library with a justification of my decision.
* **Architecture Models** – A C4 Software architecture model set of my project (including a simple UML diagram) with an explanation of the design choices made and some justification why I approached the design in that particular way.
* **UX Design** – A simple UX design, mostly consisting of wireframes with some information on my plans for the design of the final product.

# Choice of front-end language & framework

The final product’s front-end will be written primarily, as expected, in JavaScript. While the choice of language, in this case, is simple and, frankly, obvious, the choice of the framework was a little bit tougher. Some research has been done to figure out exactly which framework best matches the needs and requirements of this project. At the end of the day, I decided to pick **ReactJS** (also known as React).

## Why React?

There are many popular frameworks for JavaScript such as Vue.js, Angular, Ember, Backbone, etc. As of recently, React has gained a lot of traction and at the moment of writing, is the most popular one among them. And while React is not a framework, but a library, it still an extremely powerful tool that completely changes the basic website structure. Instead of a communication link between the HTML code and the JavaScript code using the DOM (Document Object Model), a JavaScript-based application takes the main role of the website as it generates HTML code and links all components accordingly. The following points will provide more information on the advantages of this structure:

### Time and cost-efficiency

A normal website uses the Document Object Model (DOM) – a programming interface for HTML and XML documents that connect the document structure, style and content together. Unfortunately, the Document Object Model (DOM) is slow and inefficient as it has to recalculate all CSS components, recreate the whole website layout, and essentially, remake the whole web page every time the DOM is changed. Facebook developed React to combat that inefficiency by using a virtual DOM. Unlike other frameworks that work a real DOM, ReactJS uses a copy of it that is also known as a virtual DOM. When changes are made to the virtual DOM, instead of reloading the whole page, only the changed bits of information get updated. This drastically increases the speed and cost-efficiency of the website.

### Reusable components

Similarly to Laravel, a PHP framework that was used as part of a project during semester 2, ReactJS offers the so-called components. With components, one can split the user interface (UI) into smaller, independent, and most importantly, reusable pieces of code. With a thriving community behind this popular library, many free lightweight components are available on the web. The advantage of using a popular library is the abundance of resources online that one can learn from and potentially improve upon.

## The disadvantages of React

While ReactJS has many pros, it is also important to mention some of the cons that I have taken into consideration when picking this framework/library:

* **The high pace of development** – One of the unfortunate facts of new popular framework/libraries is that they are in a constant state of flux – new features and changes are introduced regularly and old features quickly become deprecated. This is something that some developers are not comfortable with as it requires one to quickly adapt and follow its development.
* **Steep learning curve** – React is a new library (released as open-source in 2013) and pretty large. The documentation is still lacking in some sections and not polished. Without access to comprehensive official documentation, it becomes difficult to learn by yourself without depending on the available online resources – videos, community-based tutorials, etc.

# Choice of back-end language & framework

When it comes to back-end language and framework, I am still unsure as to how to continue development. The project’s RESTful API web service was built in Java using the Jersey framework. That is the main reason why I am going to stick to Java as a programming language for the time being. If I decide to rebuild my backend using another language, I will notify all other stakeholders (teachers) of this project in advance and this document will be updated accordingly. Here are the following benefits of using this language:

* **Object-oriented programming language** – Most of my programming experience has been exactly with Object-oriented programming (OOP) languages. Other than new syntax, it is not too much out of my comfort zone when it comes to using Java.
* **Low-security risk** – After some research, I found that Java is used on enterprise-level for its low-security risk. While Java is not a secure language by itself, its features save people from common security flaws. A security manager is also present – a security policy that limits users to its specified access rules.

My uncertainly with Java and the reason why I might not stay with Java as back-end programming language choice is the following:

* **Verbose and complex code** – Verbose code results in less readability. By trying to make the syntax as similar to English, the language introduces too much noise. Compared to other languages, some code pieces are much larger, which results in more points of user failure. In addition to that, it is also harder to memorize/learn Java syntax exactly due to this verbose and complex syntax structure.
* **Bad performance** – Compared to other languages, especially C and C++, it is easy to notice that it is significantly slower. At similar speed levels, languages such as Python offer a marginally better syntax and a lower learning curve. Moreover, my programming experience with Java, compared to the abovementioned language is very small.

# Architecture Models

## System Context (C1)

The system context is pretty self-explanatory. The current plans are to have two types of users (two user roles): the default user with its provided functionality and management type user that further expands the default user’s functionalities with some manager-only actions.

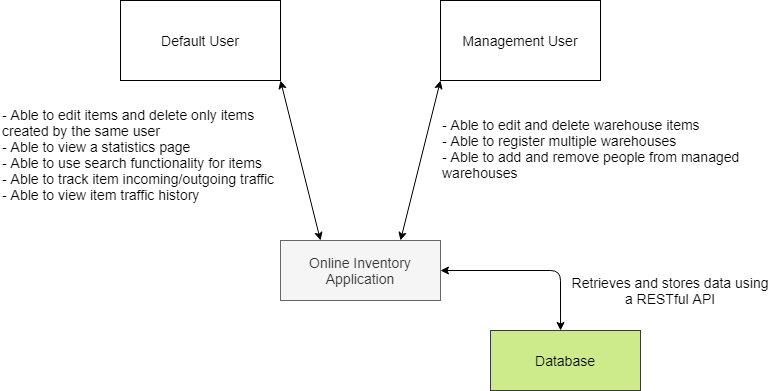


Figure 1 - System Context (C1)

## Containers and Technology (C2)

The containers and technology context provides an insight into the online inventory application’s internal structure. It is divided into two main segments: the API Application and the Web Application. The API Application will be built using Java and the Web Application will be based on JavaScript’s React library (as already mentioned in this document). Communication between these two components is done via JSON/HTTP.

Diagram

Description automatically generated

Figure 2 - Containers and technology (C2)

## Components (C3)

The components context dives further into the API application and shows the components which might form the final product. The controller is needed for user authentication for login/register. Two controllers are required to provide and summarize information when needed.

Diagram

Description automatically generated

Figure 3 - Components (C3)

## Code (C4)

The code context of the Item Mainframe shows how it will be structured to communicate with all other components of the whole application.

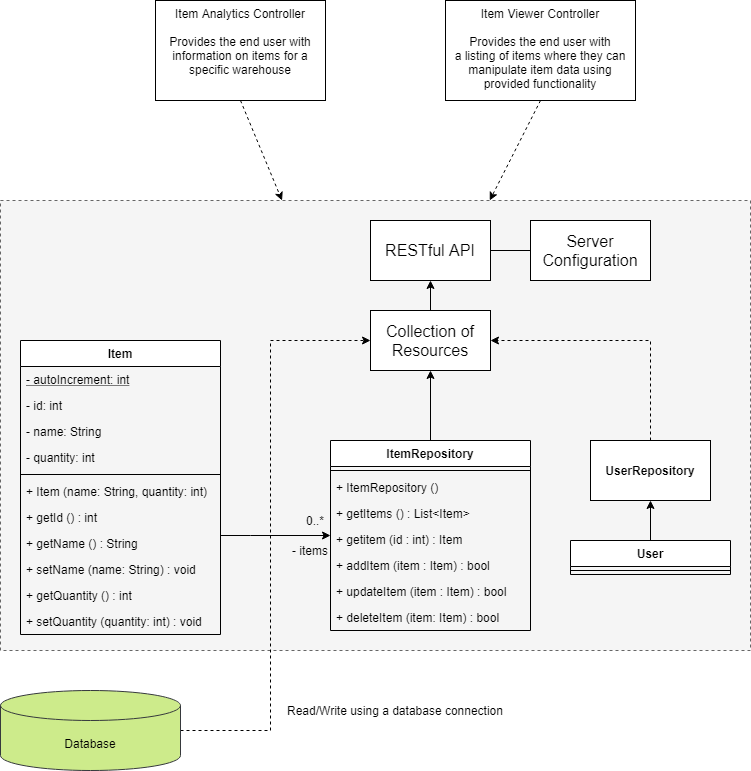


Figure 4 - Code for Item Mainframe (C4)

# Project UX Design

Graphical user interface

Description automatically generated

The goal is to provide a simple-to-use and minimalistic approach. The pages should provide only the necessary information for the given context. The colour palette should also be simple (e.g. white and an off-colour).

Once a user logs in, they will be redirected to the dashboard where they can see and manipulate item data. The page will feature two navigation bars (one for main web components, the other – item navigation and item subdirectories).

The analytics page will highlight commonly used item data, show item incoming/outgoing data, etc.

The user login/registration page will be a simple stylized form.