System Requirements Specification:

Order Management System

# Introduction

The company Company.com is in the business of e-commerce in the B2C retail market. This document relates the systems supporting order management. Before release, the Order Management System will be developed in concert with the DevOps team to tune the software, system environment and delivery process.

# Purpose

The purpose of this document is to provide details of the requirements for the Order Management System (OMS). Constraints and interactions with external systems are also presented. The document is intended primarily for the software development and DevOps teams.

# Scope

# Definitions and abbreviations

|  |  |
| --- | --- |
| Term | Definition |
| B2C | Business To Customer |
| OMS | Order Management System (system name) |
| REST | Representational State Transfer |
| API | Application Programming Interface |
| OMC | Order Management Client (product name) |
|  |  |
|  |  |
|  |  |
|  |  |

# Overall description

# Product perspective

The system is presented in figure 1 below.

A close up of a map

Description generated with very high confidence

Figure 1 – Schematic view of the system.

The user accesses the functionality via a web browser. The application servers provide the data for the client requests as JSON. The underlaying data is stored in relational databases.

# Product components

The OMS is a layered system with database, API and web client.

# Business Database

Company.com uses a relational database model for storing data related to its business. The database includes most aspects of the company’s business such as Products, Suppliers, Customers and Orders.

# Business Data API

The business data is processed and exposed by a REST API. Development of the REST API is done in Java based on the Spring Boot Framework. Maven is used as project model.

# Order Management Client

Users manage the orders via the Order Management Client (OMC). OMC is developed as a SPA using the React JavaScript library.

# Product lifecycle

The system is to be continuously developed and updated software should be continuously delivered allowing the organization to get access to the new functionality quickly.

# Communications interfaces

The user interface (OMC) communicates with the underlaying REST API using GET or POST commands.

The Business Data API server application run in a JavaEE container of an application server and conforms to the Java Servlet Specification for communication between application and server.

The Business Data API communicates with the Business Database using JDBC directly from within the application.

# Functional Requirements

# Listing of orders

The user should be able to get a listing of the all orders in form of a table.

# Filtering of orders

The user should be able to filter the listing of orders.

# Create orders

The user should be able to create a new order.

# Update orders

The user should be able to update an existing order.

# Delete orders

The user should be able to delete an order from the table by pressing delete.

# Non-functional Requirements

# Performance Requirements

The user experience for the users of OMS is important and should be a focus for both the Software Development and DevOps. The number of concurrent users is expected to be limited to less than 10.

**Response time requirements**

Must be: Less than 1.5 seconds 100% of the time.

Wish for: Less than 1.0 seconds 100% of the time.

System availability is important since order management is a business-critical activity.

**System availability requirements**

Must be: 99.95% during business hours.

Wish for: 100% during business hours.

# Security Requirements

Best practice should be used to reduce exposure to security issues.

Routines for system patching should be established.

Exclude credentials from GitHub repositories and inject those in the build and delivery process.

Be restrictive in access to file system areas related to system or system information.

**REST communication**

Pre-release: HTTP is allowed during the initial phase of the project.

Release: HTTPS is required for the communication between Order Management Client and Business Data API.

# Delivery Requirements

# Software Environment

The system is to be hosted on a Linux operating system.

Within Company.com the application of choice is Tomcat 9.

HTTP servers Apache and NGINX are allowed.

Jenkins is used for CI/CD.

MySQL is the relational database used within the company.

# Parallel Test Environment

There is a requirement for a separate test environment where new features can be explored and tested prior to being approved and deployed to the production environment. The test environment should be updated to contain the same data as the production database at the time of testing.

# Continuous Delivery

The delivery should be implemented as Continuous Delivery triggered by updated master branch that is automatically deployed to the Test Environment. The built and staged artifacts should then be easily deployable to the production environment if decided.

# Miscellaneous Details

The Business Data API should run on port 8080 and be mapped to the root of the application server.

MySQL uses the default port 3306.

# Repositories

# Business Database

GitHub: <https://github.com/devops-course-2019/business-data.git>

# Business Data API

GitHub: <https://github.com/devops-course-2019/business-data-api.git>

# Order Management Client

GitHub: <https://github.com/devops-course-2019/order-manager.git>

# Deliverables

# OMS Software Components

The different software that comprise the complete system.

# System Environment

The actual environment to be used for the described OMS applications including its delivery (CI/CD).

# System Document

Documentation of the system environment including hardware, OS, application and configuration.

# System Management Documentation

Documentation of the application management objects and how the application management should be conducted.