Contents

[1. General Information 1](#_Toc177312912)

[2. Setup 1](#_Toc177312913)

[3. Sprint 1 1](#_Toc177312914)

[3.1 IDE Setup 1](#_Toc177312915)

[3.2 Docker Containerization 1](#_Toc177312916)

# 1. General Information

# 2. Setup

Before starting with the first Sprint, following things had to be checked:

* IDE like Visual Studio or JetBrains Rider installed?
* Docker Installed?
* Git installed?

We both decided to use JetBrains Rider, since we are very familiar with their software and simply prefer using it over Visual Studio.

# 3. Sprint 1

## 3.1 IDE Setup

When creating the project, we opted for using the APS.NET Core Web API template in Rider, which provides a swagger UI out of the box. We ditched the swagger UI and just printed sample documents for Sprint 1.

## 3.2 Docker Containerization

After creating the Dockerfile for this application, the command was used to build the docker image for the container:

*docker build -t document-management-system .*

‘document-management-system’ is the name of the image.

Use this command to run the docker container:

*docker run -d -p 8081:8081 --name dms\_container document-management-system*

-p 8081:8081 lets the docker container run on port 8081 and also exposes it to port 8081, where it is being accessible in the browser. ‘dms\_container’ is the name of the container and ‘document-management-system’.

Using a *docker-compose.yml* file makes coding with Docker a lot easier. Since you can just type *docker-compose build* and *docker-compose up* to practically build and run the containerized application.