COS 497: Capstone II

Administration Manual

UMaine Athletic Department Inventory Management System

Version 1.0

Team IMSG
Collin Rodrigue, Brennan Poitras, Graham Bridges, Gabe Poulin, Sean Radel

Jude Killy - University of Maine Athletic Department

15 April 2024

UMaine Athletic Inventory Management System
Administrator Manual

Table of Contents

1. Introduction

The point of the introduction is to briefly describe the system as a whole detailing its key components, requirements, and limitations.

1.1 Purpose of This Document

This is an administrative manual for the IMSG Athletic Department Inventory Management System. This document assumes that the administrator is using at least a Windows 10 operating system. The purpose of this document is to describe the overview of our system, administrative procedures, and troubleshooting. In general, this document notes what tools are used, the background of the project, and the necessary hardware and software requirements. It also details the routine commands to launch the application using node.js and docker containers. Finally, it lists the most common errors thrown as well as known bugs and limitations. This document should serve as a comprehensive guide for future developers and administrators to effectively manage the athletic inventory. The intended readers of this document are the UMaine Athletic Department, and future UMaine COS Capstone students.

References

1. IMSG GitHub Repository "UMaine Capstone 2023-2024" April 15, 2024 https://github.com/gsb02/IMSG_Capstone
2. IMSG "Code Inspection Report" March 4, 2024 https://github.com/gsb02/IMSG_Capstone/blob/main/Artifacts/Code%20Inspection%20Review.pdf
3. Docker "Docker Manuals" April 15, 2024 https://docs.docker.com/manuals/
4. MySQL "MySQL Documentation" April 15, 2024 https://dev.mysql.com/doc/
5. Node.js "Node.js v21.7.3 documentation" April 15, 2024 https://nodejs.org/docs/latest/api/
6. React Documentation "React - The library for web and native user interfaces" April 15, 2024 https://react.dev/
7. IMSG. "System Requirements Specification" November 1 2023, https://docs.google.com/document/d/1LnOj2DEyu8DPbKXBTDBm2y6UbePr_AXC/edit
8. 2. IMSG "System Design Document" November 15, 2023, https://docs.google.com/document/d/1_dkbb3zXQzOxVbdIr3nbizQc0r-f6yC2/edit

2. System Overview

The purpose of this section is to describe an overview of the system, the software specifications to run the system, and the hardware requirements to run the system.

2.1 Background

Utilizing Docker, the system is fairly simple, requiring only a laptop or desktop computer to run the Docker container and the React app for the frontend. The system just needs to be a Windows

installation of version 22H2 or higher for Windows 10, and most recent of Windows 11. All errors and issues will be logged in Visual Studio Code (VSCode) in the terminal window the backend or frontend is run off of, so ensure to keep VSCode open.

Our program is broken into two main pieces, the frontend, and the backend, as the folders are named. The only thing needed is a terminal window linked to these two folders. These two terminal windows will log error messages and warning messages and display when things are correctly inputted into the database. Modifying files in these folders is not required and should not be conducted.

Laptops work best for this application, as the mobility a laptop provides allows for you to use the application anywhere with an internet connection. Remoting into a stationary computer is not recommended, and will not be talked about in this document.

2.2 Hardware and Software Requirements

Any Windows OS system running Windows 10, with the most recent security updates, will work for this program.

Minimum Hardware Requirements:

CPU: Any AMD or Intel processor exceeding a release date of 2018 will work
GPU: Intel UHD Graphics
RAM: 8GB
OS: Windows 10 - 22H2 or newer
HDD: 32gb or more (recommended SDD)

A user account with administrator privileges will be required to set up the environment for this application to work.

Required Software:

Docker Desktop - https://www.docker.com/products/docker-desktop/
Ensure to download and install the Windows version of Docker.

Git - https://git-scm.com/downloads
Ensure to download and install the Windows version of Git.

NodeJS - https://nodejs.org/en/download/current
Ensure to download and install the x64 Windows version of NodeJS.

Recommended Software:

This software is not necessarily required, but will make life easier for debugging and/or catching when things go wrong.

Visual Studio Code - https://code.visualstudio.com/download
Ensure to download and install the Windows version of VSCode.

Also, ensure that it is Visual Studio Code and not just Visual Studio.

3. Administrative Procedures

This section details the installation process of our code and Docker images, the routine tasks to run the containers, manage database backups, and where to seek help.

3.1 Installation

To run our program, you must first clone the GitHub repository, then install two Docker images, and finally install NPM packages.

1. Clone GitHub Repository

To clone the GitHub repository, navigate to Powershell, then run the following command
- git clone https://github.com/gsb02/IMSG_Capstone.git

2. Pull Docker Images

To pull the required Docker Images, run the following commands:
- docker image pull mysql:latest
- docker image pull node:lts-alpine3.19

3. NPM Packages

First, run the following commands to see if Node is installed
- node -v
- npm -v

If it is not installed, go to the following link and install Node.js: https://nodejs.org/en/download/

Next, in your Powershell terminal navigate to, and run the following commands:
- In the following directory: ~\IMSG_Capstone\backend, Run: npm install
- In the following directory: ~\IMSG_Capstone\frontend, Run: npm install

3.2 Routine Tasks

This section details the steps to run the Docker Containers. The Docker Containers can run the entire stack using just one command. For development purposes, we have found it is easier to run the React front-end on the local machine, rather than in a container, so that changes go live as you update them. To run the React front-end, you must first set the following environment variable in your Powershell terminal:

- $env:NODE_OPTIONS = "--openssl-legacy-provider"

To start the live React server, in your Terminal run the following command:

- npm start

Running the Docker Container

1. Starting the containers
   In your Terminal, run the following command:
- docker compose up
2. Stopping
   You can forcefully stop the container through Docker desktop, or in your Terminal with ^C, or by running the following command:
- docker compose down

3.3 Periodic Administration

It may be necessary to delete the inventory data in the system. To do so, the following steps must be taken:

1. Delete Docker Container:

First navigate to the container section, then select the “backend” container stack. Then Hit the red Delete button.

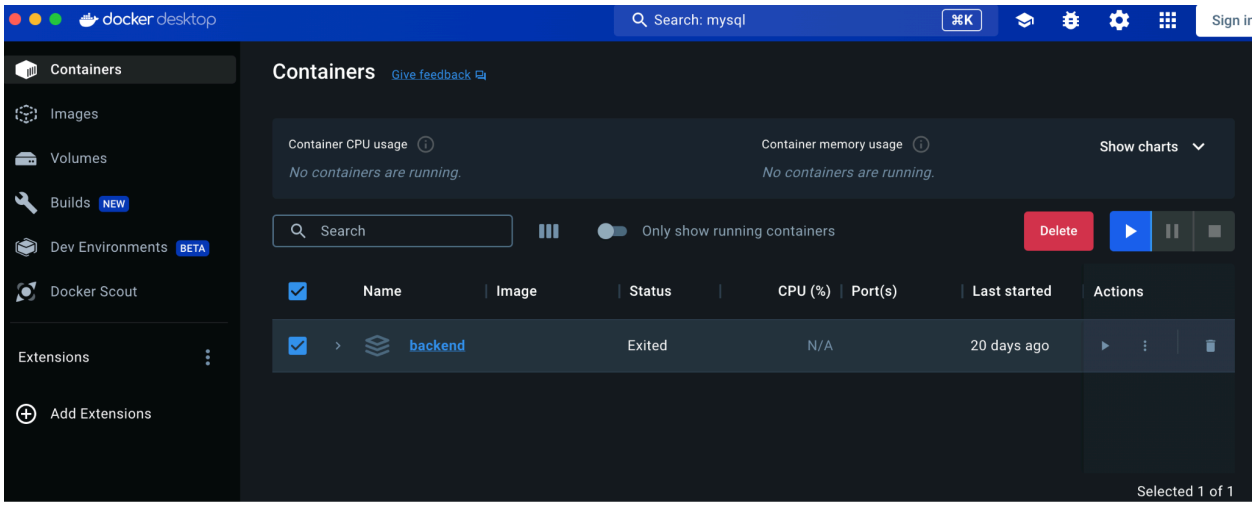

Figure 1. Selecting backend container for deletion

2. Delete Docker Volume

Next, delete the volume that contains all of the inventory system data.

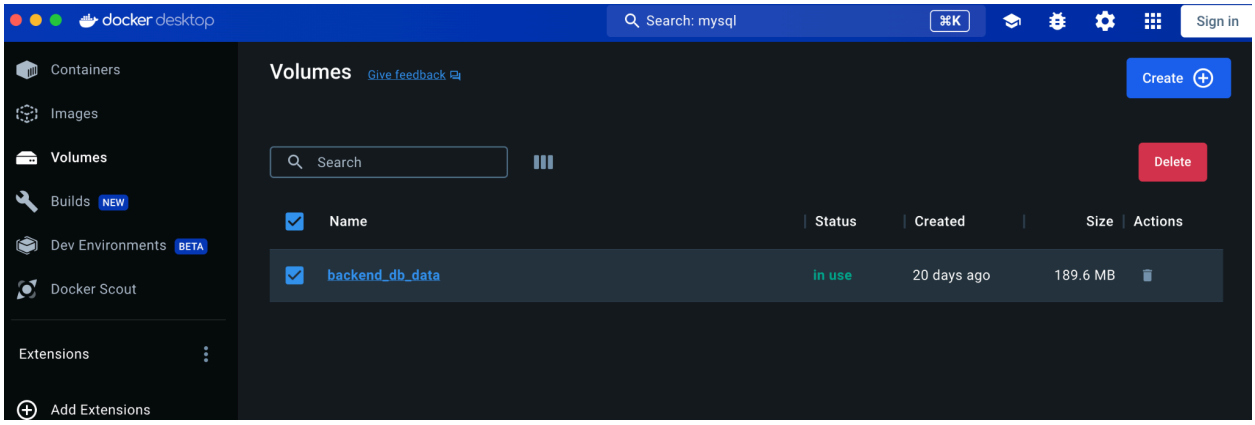

Figure 2. Deleting Volume

Following this process will reset the database, and re-running the Docker-Compose file will initiate a fresh system.

3.4 User Support

If you are looking for more information to run the system, we suggest the following resources.

1. Docker Docs
    a. https://docs.docker.com/manuals/
2. MySQL Docs
    a. https://dev.mysql.com/doc/
3. Node.js Docs
    a. https://nodejs.org/docs/latest/api/
4. React Docs
    a. https://react.dev/
5. Contact Previous Capstone Developers
    a. Sean Radel
        i. sean.radel@maine.edu
        ii. 207-604-2044
    b. Graham Bridges
        i. gsbdd02@gmail.com
        ii. (207) 251-7995
    c. Gabriel Poulin
        i. gabriel.poulin@maine.edu
        ii. 207-992-3918

    If you are continuing this project for the 2024/2025 capstone project, feel free to reach out for advice or additional support.

4. Troubleshooting

This section describes the most common errors that can be encountered in the application and actions that should be taken when these errors occur. It also includes a description of the current limitations of the application.

4.1 Dealing with Error Messages and Failures

The most common error messages in our system are typically encountered when creating and deleting objects in the system. If difficulty is encountered while performing these actions, the first suggestion is to check the console in the browser for the presence of an error message. The error codes in the console will primarily be error code 404 and error code 500.

Error Code 404: This error code means the requested action never reached the server for processing. This is typically caused if the backend is not running. If this error is received, the administrator should check the docker container to ensure the backend is running.

Error Code 500: This is the most common error encountered in our application. This error means that the information successfully reached the server but an error occurred during processing. This error could have a few different causes depending on the action being taken.

- **Creation:** If this error is encountered while creating an object it is likely the result of a mismatched data type in the input. The admin should check that text fields contain only alphanumeric characters. The data in the form should match the formats provided in the user guide.

- **Deletion:** If this error is encountered while deleting an object, it is likely that the object still has existing relationships in the system. For example, a player cannot be deleted from the system unless the player's assigned equipment is deleted first. This is typically handled automatically by the system but in some cases the user may be required to perform the actions manually.

4.2 Known Bugs and Limitations

The largest known limitation is sensitivity to data types during information input. If the data types do not match the database specification, the request will fail. Users should ensure the data inputs match the formats that will be provided in the user guide.

A major limitation to our current implementation of MySQL and Docker is that when the database schema needs to be updated, all data in the database is lost. This is due to the fact that the container and volumes need to be deleted, and the container's need to be redeployed, for the schema changes to take place. We are not currently aware of a better way to update the database schema in our code without refreshing the entire system.

The security of the system is a large limitation to the application being deployed. The application should only be run locally because it provides no user authentication and input fields may be vulnerable to SQL injection and scripting attacks.

Our project Code Inspection Review Document (Reference 2) describes all known coding defects within our system.

Appendix A – Team Review Sign-off

Team IMSG has thoroughly reviewed the Administration Manual document for the Athletic Inventory System and has agreed that the following information is accurate. Collectively we have no major contentions in the information stated in the document. By signing this agreement, one acknowledges all the terms and conditions outlined in the document and understands the importance of effective team collaboration, communication, and shared accountability when achieving the goals of the project. By signing below, we pledge our dedication to the success of the team and the project we plan to undertake. We agree to work collaboratively, and support each other to uphold the guidelines and expectations of this project.

Signature:	Date:	Printed:
X: *Collin Rodrigue*	4/14/2024	Collin Rodrigue
X: *Gabriel Poulin*	4/14/2024	Gabriel Poulin
X: *Brennan Poitras*	4/14/2024	Brennan Poitras
X: *Sean Radel*	4/14/2024	Sean Radel
X:*Graham Bridges*	4/14/2024	Graham Bridges

Appendix B – Document Contributions

Name	Date	Contribution	Version
Gabe Poulin	4/14/2024	System Overview	1
Sean Radel	4/14/2024	Administrative Procedures	1
Brennan Poitras	4/15/2024	Troubleshooting	1
Collin Rodrigue	4/15/2024	Introduction	1
ALL	4/15/2024	Appendix A,B	1