**Cork Institute of Technology**

**High Level Design Document**

**Cloud Development Frameworks Assignment 2**

**Jimmy Collins – R00145569**

Contents

[Section 1 - Introduction 2](#_Toc482630592)

[Section 2 - Microservices Overview 3](#_Toc482630593)

[Cart Service 3](#_Toc482630594)

[Purpose 3](#_Toc482630595)

[Functionality Provided 3](#_Toc482630596)

[Catalogue Service 5](#_Toc482630597)

[Purpose 5](#_Toc482630598)

[Functionality Provided 5](#_Toc482630599)

[User Service 6](#_Toc482630600)

[Purpose 6](#_Toc482630601)

[Functionality Provided 6](#_Toc482630602)

[Stock Administration Service 7](#_Toc482630603)

[Purpose 7](#_Toc482630604)

[Functionality Provided 7](#_Toc482630605)

[Order Service 9](#_Toc482630606)

[Purpose 9](#_Toc482630607)

[Functionality Provided 9](#_Toc482630608)

[Front-end Service 11](#_Toc482630609)

[Purpose 11](#_Toc482630610)

[Functionality Provided 11](#_Toc482630611)

[Section 3 – Database Design 12](#_Toc482630612)

# Section 1 - Introduction

This document outlines the design of the ‘*Super Carz Inc.*’ site that that I decided to implement for my final project. The code base is based on the Assignment 1 code base, with the necessary improvements and additions required to make the application functionally complete, and some user interface changes to improve the aesthetics of the site.

The main purpose of the site I have implemented is to allow users to view and purchase high-end cars. It is comprised of a number of services, each of which is outlined in depth in Section 2 of this document.

The application is currently hosted on IBM Bluemix and is available at <https://shop-jc.mybluemix.net/> for viewing[[1]](#footnote-1).

To use the site at the above link, the following Administrator and Non-Administrator accounts can be used. The Administrator account can access functions like Stock Management etc.

**Administrator**

* Username: jim
* Password: jim

**Non-Administrator**

* Username: joe
* Password: joe

Note that I have also compiled a developer journal for this project that outlines my efforts in the development and deployment of the project. This will be submitted along with the full codebase before the May 22nd deadline.

# Section 2 - Microservices Overview

This section outlines the operation of each of the services that comprise the ‘*Super Carz*’ application.

There are 6 in total:

* Cart Service
* Catalogue Service
* User Service
* Stock Service
* Order Service
* Front-end Service

## Cart Service

### Purpose

The purpose of the cart service is to provide an (in-memory) storage area for items that a customer has added to their cart on the site which they may wish to purchase later when they use the checkout functionality.

The cart is mainly used by the front-end service to provide this functionality.

### Functionality Provided

The cart service provides three API’s which are outlined below.

HTTP POST

#### /add

This API provides the ability to add a specified quantity of an item to a customer’s cart. The item is identified by a unique product ID. If an item with this product ID already exists in the cart, then the items will be combined. As well as each item being identified by a product ID in the cart, it is also uniquely identified by a cart ID.

#### Example JSON Payload



HTTP GET

#### /cart/:custId/items/

This API provides the ability to retrieve the current contents of a specific customer’s cart as JSON.

#### Example JSON Response



HTTP DELETE

#### /cart/:custId/items/:id

This API provides the ability to delete an item from the customer’s cart.

For example, /cart/5/items/1 would delete item 1 in the cart of the customer with the customer ID of 5.

## Catalogue Service

### Purpose

The catalogue service handles interactions with the backend MySQL database where details of the available products are stored (*high-end cars in the example of the site that I have developed based on the Assignment 1 Codebase*).

The functionality within the catalogue service is used by both customer facing features and some of the administrator-only features.

### Functionality Provided

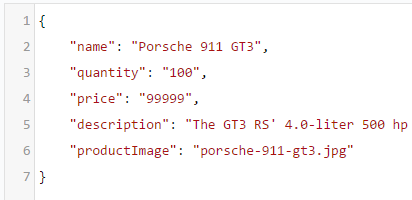
The following functionality is provided within the catalogue service.

HTTP POST

#### /newProduct

This API provides the ability to add new products to the catalogue. This area from which this functionality is available in the front-end service is restricted to administrator users.

#### Example JSON Payload



#### /deleteProduct

This completely deletes a product from the product catalog. It is an irreversible action.

#### /deactivateProduct

This provides the ability to deactivate a product in the product catalog. This does not delete a product, but makes it inactive in the database such that it will no longer appear for sale on the site. This function is only accessible from the Stock Management UI in the Administration Panel.

#### /reactivateProduct

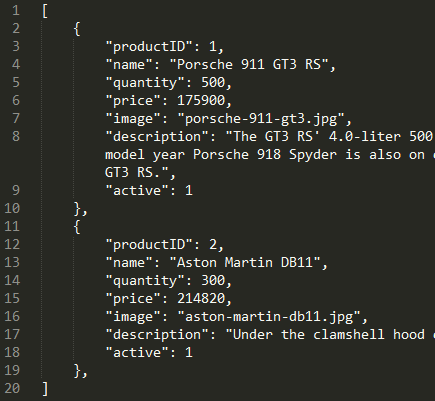
This is the reverse of the above, and can be used to reactivate a product on the site and make it available for sale again. This is also only accessible from the Stock Management UI in the Administration Panel.

HTTP GET

#### /getProducts

This API provides the ability to get all the active products in the catalog. This is used by the front-end service to show the available products to users.

#### Example JSON Response



## User Service

### Purpose

The user service is responsible for user management with regards to user registration and handling requests to login to the application.

### Functionality Provided

This service provides the following API’s that are used by the front-end service for user registration and user login validation.

HTTP POST

#### /login

The registration user interface on the front-end calls into this API in order to add new users to the users table in the database. All new users are added as Customer users, but any user can be made an Administrator later in the database.

TODO – Information on Cookies?

The registration functionality will return errors in the event of a user already existing in the database (based on the username), or if any MySQL errors occur.

#### /register

This logic is invoked from the front-end service and is responsible for validating that user credentials are correct when a user attempts to login to the application. If the credentials are not valid, then an error is returned to the front-end service which displays a message to the user.

If successful, the customer ID and customer type are returned to the front-end service as a JSON response.

## Stock Administration Service

### Purpose

The purpose of this service is to provide a set of API’s that can be used (via the front-end service user interface), to administer stock levels within the application.

### Functionality Provided

The following functionality is available in this service.

HTTP POST

#### /incrementStock

The ability to increment stock levels of a particular item from the catalogue. This would be used for example in the event of new stock being purchased by the company.

#### Example JSON Payload



#### /decrementStock

The ability to decrement stock levels for a particular item from the catalogue. This would be used in the event of a customer making a purchase of this item.

#### Example JSON Payload

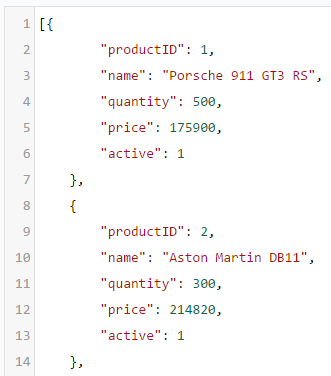


HTTP GET

#### /currentStock

The ability to get the current stock levels for each product. This is used in the Stock Management user interface in the administration panel to show an overview of the stock levels and value for all current products in stock.

#### Example Response



## Order Service

### Purpose

This service is used to process user orders. It is quite simplistic currently in its implementation in that it currently does not handle payments, just the recording of user orders.

### Functionality Provided

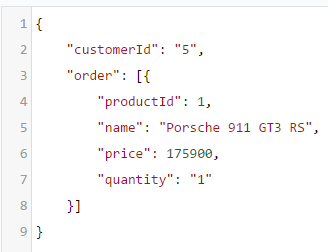
The following functionality is available in this service.

HTTP POST

#### /order

The ability to place an order for a customer. In the database a record of this order is persisted. When an order is placed, the stock level for the particular items on order will be decremented automatically.

#### Example JSON Payload

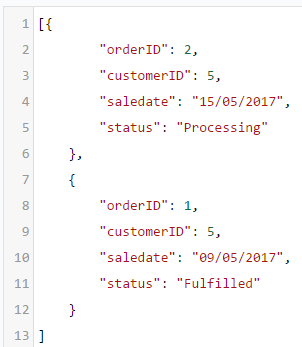


HTTP GET

#### /order

The ability to retrieve a list of orders for a particular user. Users can view their historical orders via a link in the navigation bar when the user is logged in to the application.

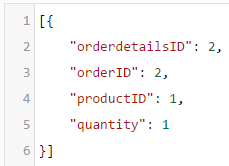
#### Example JSON Response



#### /orderDetails

This is used to view the low level details of a particular order. This data is accessible to both users and administrators via an option in the user interface.

#### Example JSON Response



#### /allOrders

This API will return all historical orders for all customers in every state. It is a function that is only accessible to Administrator users from the Administration panel. The JSON response is similar to that of the /order API.

## Front-end Service

### Purpose

The front-end service is responsible for defining and managing the main user interface of the application. It is to main entry point to the application for users. In order to achieve this, it interacts with each of the other services.

Within the front-end source tree, the ‘api’ folder contains wrappers that enable the front-end to talk to the other services.

The cart, catalogue, user, stock, and order service need to be up and running and accessible to the front-end service in order for it to function correctly.

### Functionality Provided

The front-end services provides the user interface definition, and integrates with the other services in order to provide the application functionality.

In contrast to the previous services discussed in this document which don’t have user interfaces and just provide API’s, the front-end service contains the mark-up that defines the entire user interface of the application.

The user interface is defined in public/index.html.

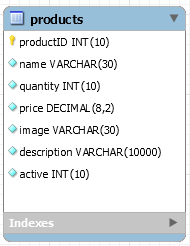
Most of the user interface styling is accomplished via bootstrap[[2]](#footnote-2), but there is a separate style.css in the same directory which defines some custom CSS.

# Section 3 – Database Design

The database design is quite simple currently, containing just 4 tables that handle the persisting the required data.

## Products Table

The Products table is used to store the items that are available for sale on the site.

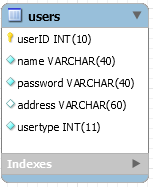


Column Definitions

TODO

## Users Table

The users table TODO

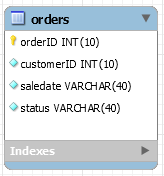


Column Definitions

TODO

## Orders Table

TODO

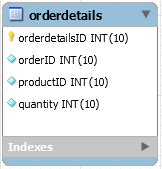


Column Definitions

TODO

## OrderDetails Table

TODO



Column Definitions

TODO

1. Note that the application is still in development at time of writing this document, please ignore any issues/downtime until the final codebase is submitted before the Assignment 2 deadline (May 22nd). [↑](#footnote-ref-1)
2. <http://getbootstrap.com/> [↑](#footnote-ref-2)