# ISAD251 Database Application Development

## Application Fact Sheet

20 CREDIT MODULE / 50% COURSEWORK SUBMISSION

/ 50% EXAM

MODULE LEADER: SHIRLEY ATKINSON

MODULE TUTOR: MARTIN READ

CRAIG BANYARD

MODULE AIMS This module aims to introduce students to the concepts and issues concerning server-side applications interfacing multi-user, networked, relational databases and to provide a solid foundation in SQL

ASSESSED LEARNING OUTCOMES (ALO):

1. Write effective SQL statements for defining, manipulating and controlling data.

2. Design and implement a multi-user database application

3. Implement server-side web solutions using appropriate technologies that integrate with back-end data stores

4. Design and implement applications providing and consuming a distributed API

# Requirements

## Functional Requirements

The basic functional requirements for the application have been provided in the following user stories:

As a customer I wish to order a drink/snack.

As a customer I wish to see what I have ordered.

As a customer I wish to add to my current order for a drink/snack.

As a customer I wish to cancel my order for a drink/snack.

As the admin I wish to enter details of the drinks/snacks I have for sale.

As the admin I wish to read the details of the drinks/snacks I have for sale.

As the admin I wish to view a customer’s order(s).

As the admin I wish to edit the details of the drinks/snacks I have for sale.

As the admin I wish to withdraw a drink/snack from sale.

## Non-functional Requirements

These requirements will be characteristics of the system that are not described above.

### Technical Requirements:

The technological stack for development that I have chosen for this application will involve ASP.NET and Microsoft SQL Server.

The application will run on a web server provided and will be written in ASP.NET.

The database will be remotely hosted on socem1 and use Microsoft SQL Server.

The interface that both the customer and user will access the application through will be via a desktop browser.

### Performance Requirements:

This application will be running on the network set up within the university and therefore the speed of the application is not relevant to this task as it is beyond the scope of the application.

### Usability Requirements:

The application and GUI’s used will conform to all accessibility rules as per the W3 validator.

### Reliability Requirements:

Reliability issues are outside of the scope of this application and are primarily dependent on the structure of the labs and servers. However, in terms of data reliability, all data added/edited/deleted will be updated to the created database when using the application.

# Planning

In this section, the user stories provided have been abstracted and simplified to plan out how the users of the application will navigate through the application and how the interface should look. During this process, it is important to refer to the functional requirements specified for each user story and ensure that all requirements are met by the pseudocode and storyboards.

## Pseudocode for user stories:

### Customer user story:

1. Customer opens application at main page
2. Customer clicks button on page to access menu
3. While customer hasn’t clicked view basket button
   1. Customer chooses item to add to basket
   2. Customer chooses quantity of item requested
4. Customer clicks button to view basket of items
5. Displays to user all selected items, quantity and price
6. Customer can add, delete and edit items in order
7. Customer clicks button to complete or cancel order
8. If complete order = clicked
   1. Confirmation of order is displayed to user
9. If cancel order = clicked
   1. Confirmation of cancel order is displayed

### Admin user story:

1. Admin opens application at admin page
2. Admin can click buttons for:

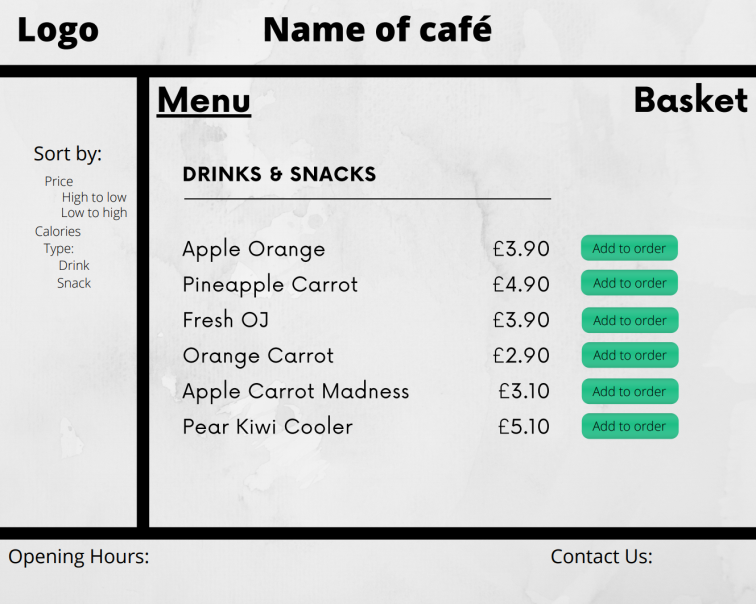
* View products
* View orders

1. If view products = clicked
   1. Display to admin table of products
   2. Admin can click on buttons to add, edit and delete products
   3. Each button will bring up appropriate form for admin to fill out essential details
   4. Admin hits confirm button
   5. Confirmation of change is shown to admin
2. If view orders = clicked
   1. Display to admin table of orders

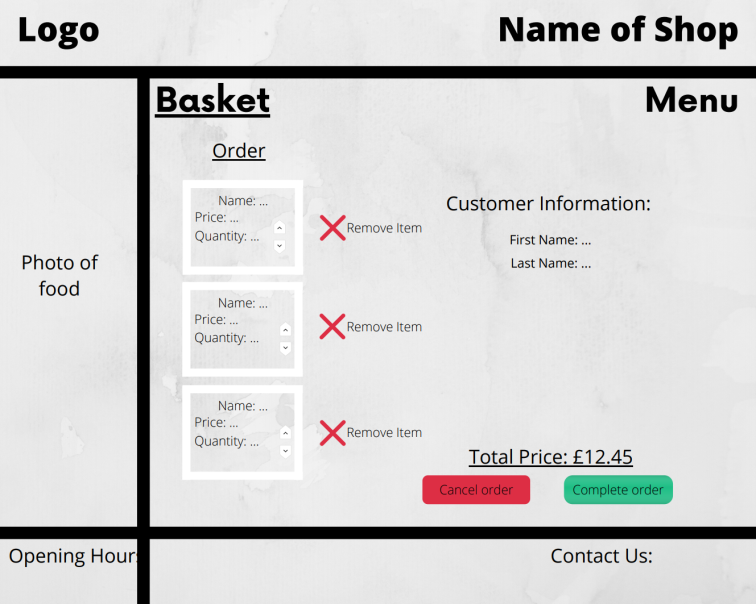
## User Storyboards

This storyboard allows us to view the website from an end user’s perspective. This is beneficial in the planning process as it will highlight the usability aspects of the website and how the end user will interact with it. In terms of developing the storyboard, it is always important to relate to the functional requirements of the application to help create a clean and concise application that specifically meets all of the functional requirements specified.

### Customer User storyboard

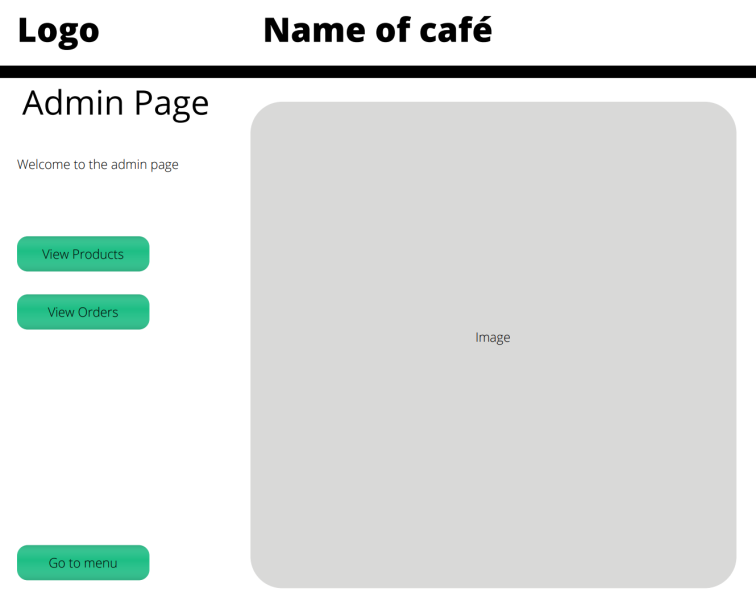


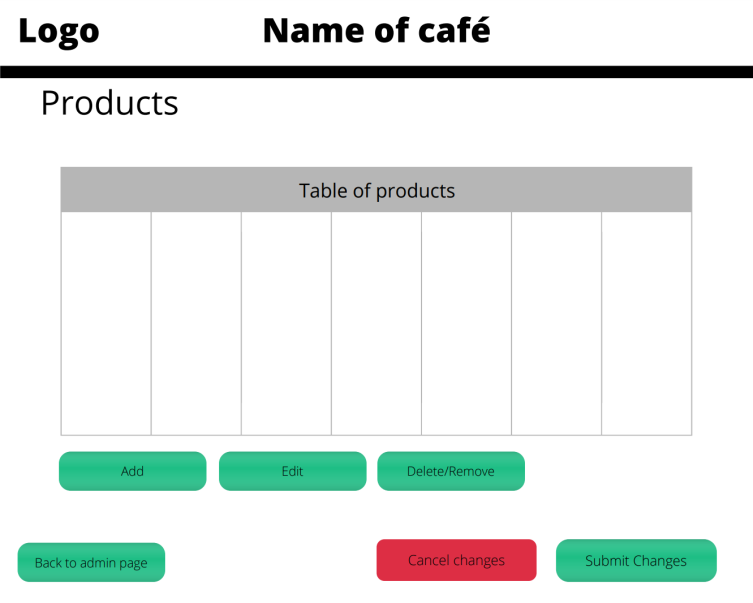
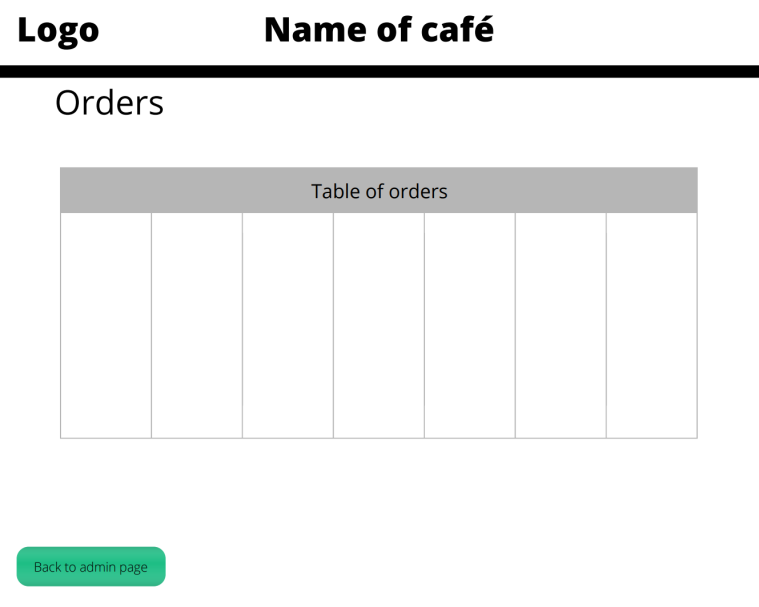
Index.php



Basket.php

### Admin User storyboard



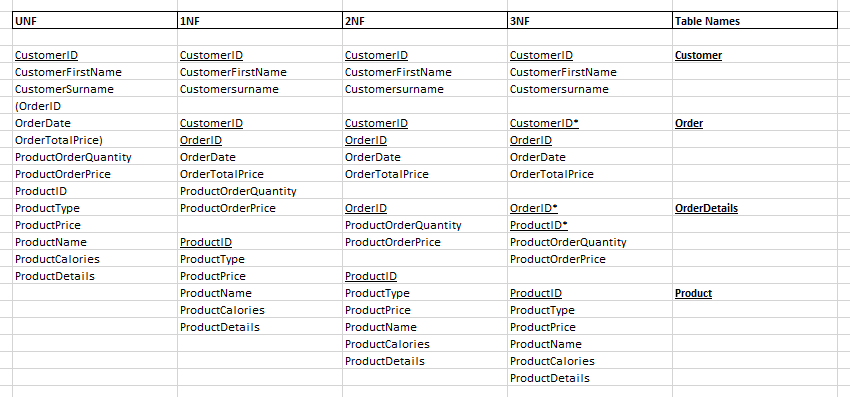


# Database Design

## Normalisation

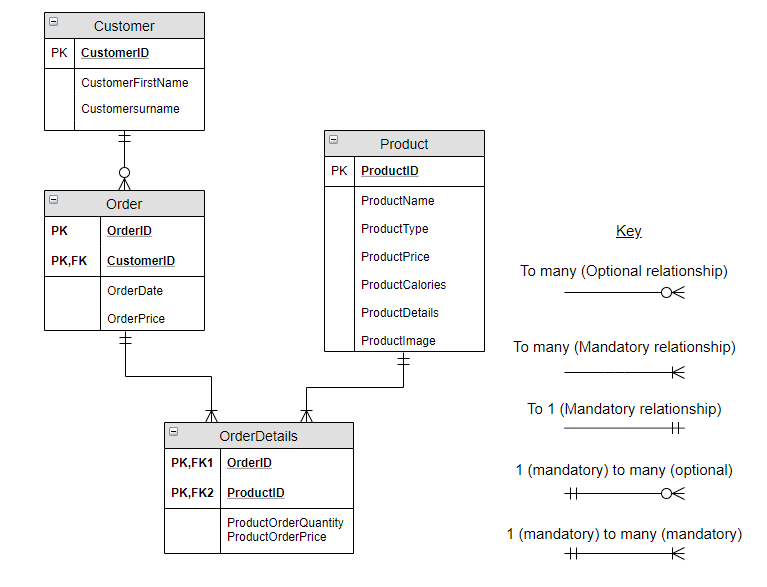
This process will allow the database to accord to the normal forms used in a relational database. This will significantly reduce the data redundancy and also improve the integrity of data within our relational database.

This normalisation will map back to the elements shown in the user stories and requirements.



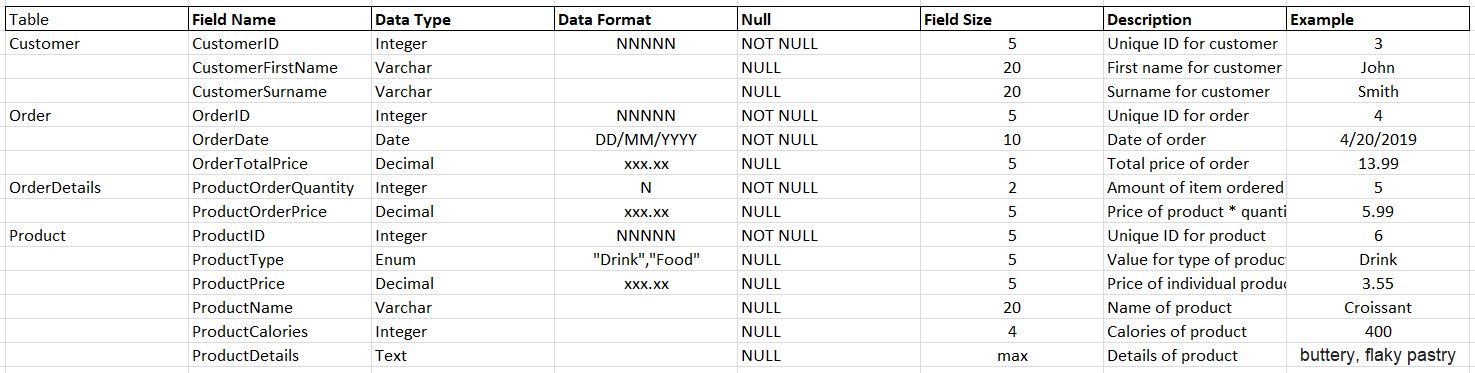
## Entity Relationship Diagram

The entity relationship diagram highlights the data aspect of our application. This structural diagram will aid the visualisation of the database design by identifying the major entities and the inter-relationships among these entities.



## Data Dictionary

This data dictionary provides a well-defined description of the contents, format and structure of the database.



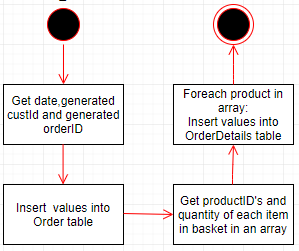
## UML Diagrams

UML diagrams help to model the structure and behaviour of the application as a whole. By creating activity diagrams for the three-layers of the application we will be able to gain a clearer understanding on the architecture and relationships between classes.

### Stored procedures

These activity diagrams represent the stored procedures that will be used in order to achieve the functionality requirements.

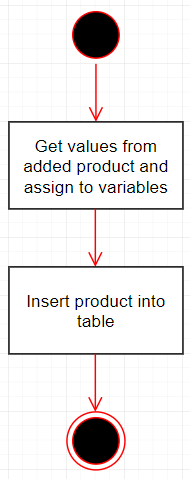
|  |  |
| --- | --- |
| Operation name | Enter\_Order |
| Involved tables | Customer, Order, OrderDetails |
| Overview | When a customer confirms an order |
| Pre-condition | All tables involved have been created |
| Post-condition | A new record is added into the order table |
| Input parameters | @Datetime, @Total price, @CustomerID, [@ProductID, @Quantity] for as many products in order |
| Output parameters | None |
| Return value | None |



“As a customer I wish to order a drink/snack.”

This stored procedure maps back to the customer user story and will help to meet the functional requirements specified in the brief.

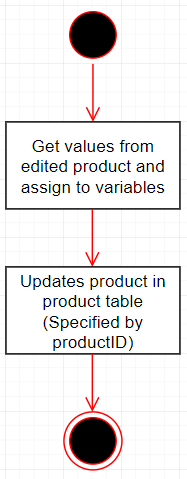
|  |  |
| --- | --- |
| Operation name | Add\_Product |
| Involved tables | Product |
| Overview | When an admin adds a product |
| Pre-condition | Product table has been created |
| Post-condition | A new record is added into the product table |
| Input parameters | @ProductType, @ProductPrice, @ProductName, @ProductCalories, @ProductDetails |
| Output parameters | None |
| Return value | None |



“As the admin I wish to enter details of the drinks/snacks I have for sale.”

This stored procedure maps back to the admin user story and will allow the admin to add a new product to the products table.

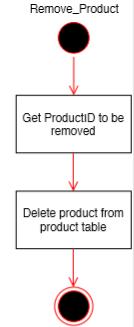
|  |  |
| --- | --- |
| Operation name | Edit\_Product |
| Involved tables | Product |
| Overview | When an admin edits a product |
| Pre-condition | Product table has been created and at least 1 product exists in the table |
| Post-condition | A new record is added into the product table |
| Input parameters | @ProductID, @ProductType, @ProductPrice, @ProductName, @ProductCalories, @ProductDetails |
| Output parameters | None |
| Return value | None |



“As the admin I wish to edit the details of the drinks/snacks I have for sale.”

This stored procedure maps back to the admin user story and will allow the admin to edit existing products in the products table.

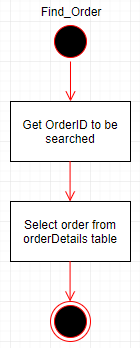
|  |  |
| --- | --- |
| Operation name | Remove\_Product |
| Involved tables | Product |
| Overview | When an admin withdraws a product |
| Pre-condition | Product table has been created and at least 1 product exists in the table |
| Post-condition | An existing record is deleted from the table |
| Input parameters | @ProductID |
| Output parameters | None |
| Return value | None |



“As the admin I wish to withdraw a drink/snack from sale.”

This stored procedure maps back to the admin user story and will allow the admin to remove existing products in the products table.

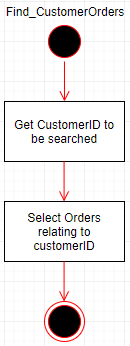
|  |  |
| --- | --- |
| Operation name | Find\_Order |
| Involved tables | Order |
| Overview | When an admin wants to find the products bought in a specific order |
| Pre-condition | Order table has been created and at least 1 order exists in the table |
| Post-condition | Products and quantity in order are displayed to admin |
| Input parameters | @OrderID |
| Output parameters | None |
| Return value | None |



“As the admin I wish to view a customer’s order(s).”

This stored procedure maps back to the admin user story and will allow the admin to search for a specific orders details.

|  |  |
| --- | --- |
| Operation name | Find\_CustomersOrders |
| Involved tables | Order |
| Overview | When an admin wants to view a specific customers’ orders |
| Pre-condition | Order table has been created and at least 1 order exists in the table |
| Post-condition | All of customer’s orders are displayed to user |
| Input parameters | @OrderID |
| Output parameters | None |
| Return value | None |



“As the admin I wish to view a customer’s order(s).”

This stored procedure maps back to the admin user story and will allow the admin to search and view all of a specific customer’s orders.

### Views

The data layer of the application will contain several database views that will help to meet the functionality requirements specified.

“As the admin I wish to view a customer’s order(s). “

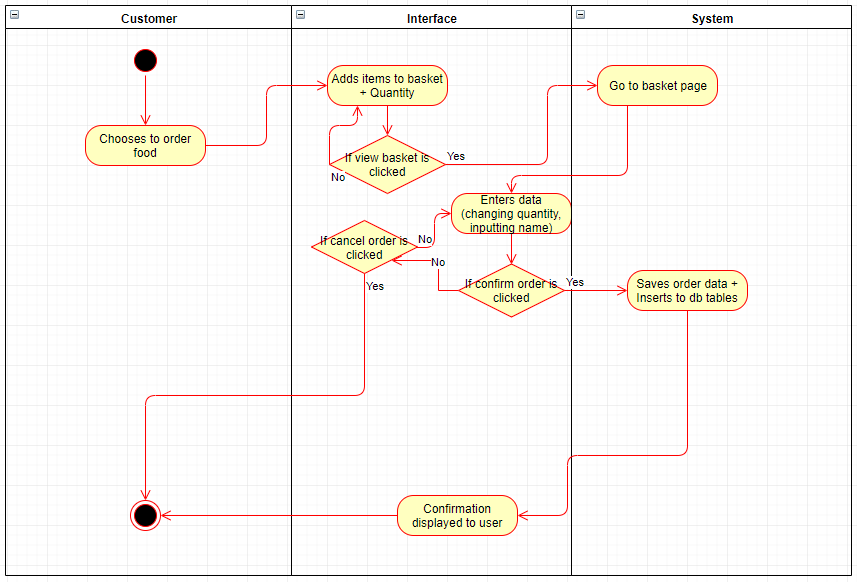
|  |  |
| --- | --- |
| View name | View\_Orders |
| Involved tables | Order |
| Involved fields | OrderID, OrderDate, OrderTotalPrice |
| Overview | Displays all orders to admin |
| Pre-condition | Order table created and contains existing records |
| Post-condition | Outputs data from order table |

“As the admin I wish to read the details of the drinks/snacks I have for sale. “

|  |  |
| --- | --- |
| View name | View\_Products |
| Involved tables | Products |
| Involved fields | ProductID, ProductName, ProductType, ProductPrice, ProductCalories, ProductDetails |
| Overview | Displays all current products to admin |
| Pre-condition | Product table created and contains existing records |
| Post-condition | Outputs data from product table |

### Activity Diagram

The activity diagram is a behavioural diagram that will show the flow of the application from a specified start and finish point.



### Sequence Diagram

The sequence diagram models object interactions in a consequential sequence. It is used to show the order of which the objects and classes involved are executed in the workflow of the entire scenario/process.

# API

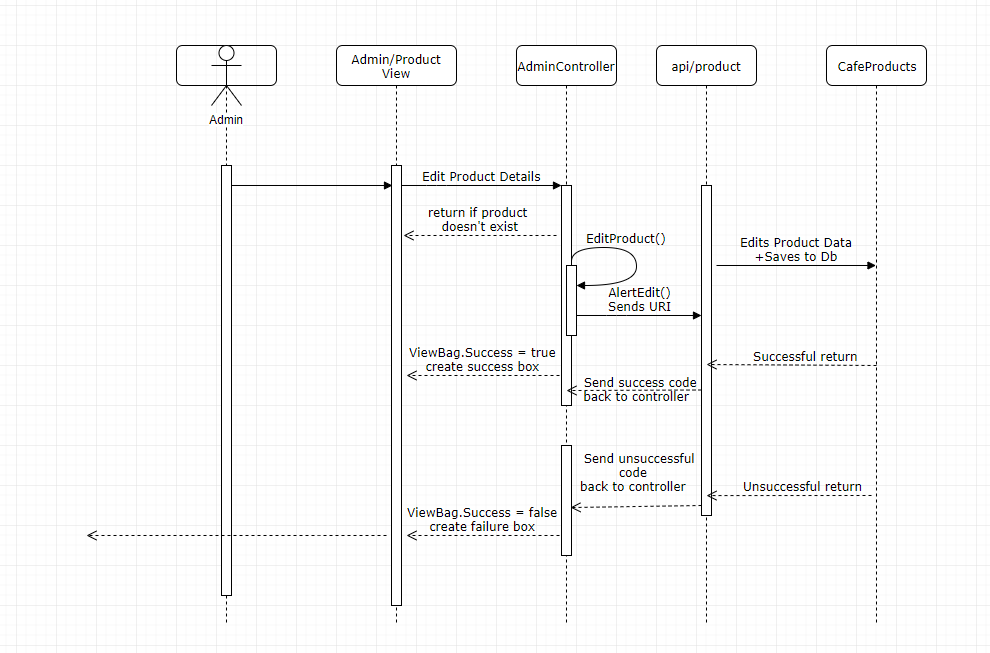
The API that I have created for this application exposes the “products” and “customers” resources. They will be used to interact with the database to allow the admin to edit, create and delete data in the order and customer tables within the database. This will be done by using the HTTP verbs PUT (Edit) POST (Create) and DELETE (Delete) in the admin controller of the MVC application and will save all changes made by the admin to the database.

The sequence diagrams below show how the API will be used in the admin story board to meet the functional requirements for the admin.

“As the admin I wish to edit the details of the drinks/snacks I have for sale.”

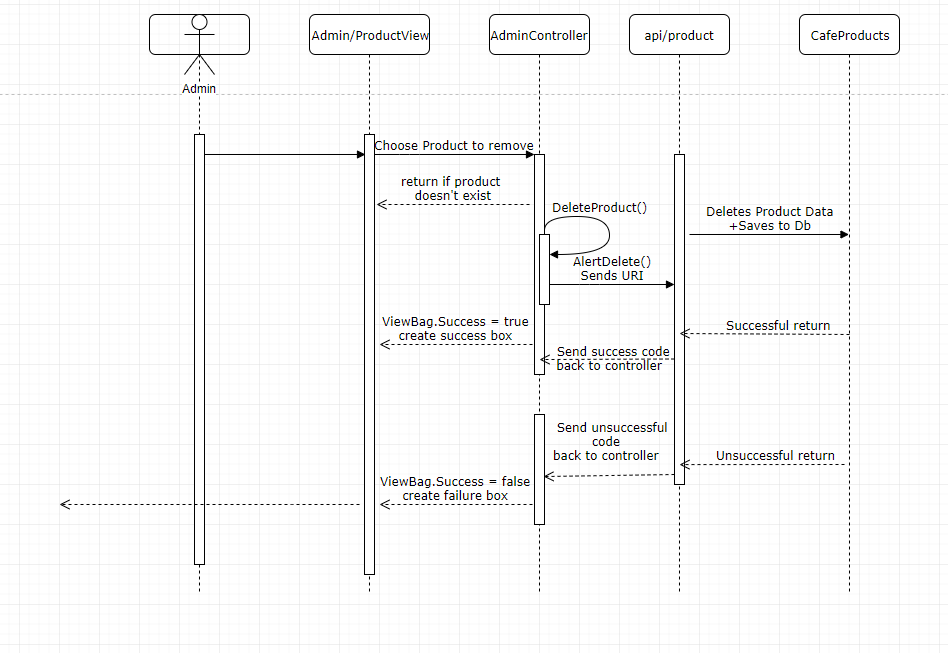
“As the admin I wish to enter details of the drinks/snacks I have for sale.”

This process will use the HTTP verb PUT to edit the specified product.



“As the admin I wish to withdraw a drink/snack from sale.”

This process will use the HTTP verb DELETE to delete the specified product.



Similar functionality has also been used to allow the admin to edit the customer and also delete customers from the database using the API.