PRCS252 – Travel Management System: Final Report

William Butler, Andrew Bellas, Vincent Castellani and Goel Biju

07 May 2019

Contents

[2 Project Management 5](#_Toc8105991)

[2.1 Team Project Management 5](#_Toc8105992)

[2.2 Changes during the Project 5](#_Toc8105993)

[2.3 Risk Analysis 5](#_Toc8105994)

[2.4 Difficulties during the project 6](#_Toc8105995)

[3 Requirements Analysis and Design 7](#_Toc8105996)

[3.1 Product Backlog 7](#_Toc8105997)

[3.2 Use Case UML Diagrams 11](#_Toc8105998)

[3.2.1 Administrator 11](#_Toc8105999)

[3.2.2 Driver 12](#_Toc8106000)

[3.2.3 Customer 13](#_Toc8106001)

[3.2.4 Manager 14](#_Toc8106002)

[3.3 Use Case Realisation 15](#_Toc8106003)

[3.3.1 Updating a Route 15](#_Toc8106004)

[3.3.2 Booking onto a Journey 17](#_Toc8106005)

[3.4 Sprint Plan 19](#_Toc8106006)

[3.4.1 Sprint 1 19](#_Toc8106007)

[3.4.2 Sprint 2 20](#_Toc8106008)

[3.4.3 Sprint 3 21](#_Toc8106009)

[3.4.4 Sprint 4 23](#_Toc8106010)

[3.5 Changes During Development 24](#_Toc8106011)

[4 Database Design 25](#_Toc8106012)

[4.1 Database Diagram 25](#_Toc8106013)

[4.2 SQL Create Statements 26](#_Toc8106014)

[4.2.1 Employee Table 26](#_Toc8106015)

[4.2.2 Employees Archive 28](#_Toc8106016)

[4.2.3 Shifts 30](#_Toc8106017)

[4.2.4 Customers 32](#_Toc8106018)

[4.2.5 Stops 35](#_Toc8106019)

[4.2.6 Routes 36](#_Toc8106020)

[4.2.7 Routes Archive 37](#_Toc8106021)

[4.2.8 Route Stops 37](#_Toc8106022)

[4.2.9 Coaches 38](#_Toc8106023)

[4.2.10 Coaches Archive 39](#_Toc8106024)

[4.2.11 Journeys 40](#_Toc8106025)

[4.2.12 Bookings 43](#_Toc8106026)

[4.2.13 Booking History 45](#_Toc8106027)

[4.2.14 Replacements 47](#_Toc8106028)

[4.3 Normalised Tables 48](#_Toc8106029)

[4.3.1 First Normalised Tables 48](#_Toc8106030)

[4.3.2 Second Normalised Tables 49](#_Toc8106031)

[4.3.3 Class Diagram Comparison to Normalised Tables 50](#_Toc8106032)

[5 Usability 50](#_Toc8106033)

[5.1 Cognitive Walkthrough Analysis 50](#_Toc8106034)

[5.1.1 Outline 50](#_Toc8106035)

[5.1.2 User evaluation feedback priority list (in order of feedback received and changes needed, gained from feedback) 50](#_Toc8106036)

[5.1.3 Usability Test of the Mobile App 54](#_Toc8106037)

[6 Security 56](#_Toc8106038)

[6.1 Features Implemented 56](#_Toc8106039)

[6.1.1 SQL Injection 56](#_Toc8106040)

[6.1.2 Hashing and Salting 56](#_Toc8106041)

[6.1.3 Token based Authentication 56](#_Toc8106042)

[6.1.4 Role-based authentication 57](#_Toc8106043)

[6.2 Features Not Implemented 57](#_Toc8106044)

[6.2.1 HTTPS 57](#_Toc8106045)

[6.2.2 Data in Transit – API 57](#_Toc8106046)

[7 Software Engineering 58](#_Toc8106047)

[7.1 Use of Software Patterns 58](#_Toc8106048)

[7.1.1 MVC Pattern 58](#_Toc8106049)

[7.1.2 Singleton Pattern: 58](#_Toc8106050)

[7.2 Good Areas 58](#_Toc8106051)

[7.2.1 API Logic 58](#_Toc8106052)

[7.2.2 Object Mapping 58](#_Toc8106053)

[7.2.3 Expandability 59](#_Toc8106054)

[7.3 Shortcomings of the System 59](#_Toc8106055)

[7.3.1 Multi-user authentication 59](#_Toc8106056)

[7.3.2 Additional Functionality 60](#_Toc8106057)

[7.3.3 Code Reviews and Testing 60](#_Toc8106058)

# Project Management

## Team Project Management

In terms of the project management once all the user stories were decided and organised into sprints, each member of the team would choose an equal number of users. When possible a user story for each application would be chosen, this was done to equally spread the workload of the applications.

## Changes during the Project

Further into the project there was backlog that had to be considered with each sprint. To deal with the backlog certain user stories were prioritised to ensure that there is a higher chance of achieving the minimum viable product. These changes had to take place because of certain parts of development taking longer than expected, for example the API.

All of the applications had changes to the user interface. The website was first designed by manually coding html, CSS and JavaScript in visual studio. It was then later realised that visual studio could auto generate scaffolded webpages. This was opted for as it created a more consistent website and allowed the database easier to retrieve. The design of the mobile application was altered as when it was designed the limitations of the Android Studio software was not known. There was also changes made that took the cognitive walkthrough results into consideration. The desktop application had the most change from its original design. The initial design plan was not consistently followed and further into the project it was decided to condense the application and whittle it down to as minimum as possible as the driver would not need a mass of information to do their job.

## Risk Analysis

Looking back at our risk analysis there are some that standout, the first being developing something too technically difficult. The initial ideas were quite ambitious and once we had gone through the requirements and figuring out what needed to be done it became obvious that some functionality needed to be left to the side and the core of the applications should be the biggest focus. There was research done into the software that was used to ensure that it would be able suitable for what we needed it for. There was still a big knowledge gap on how to use the software effectively, which did slow down progress at the beginning of the project.

Another risk was developing the wrong user interface, the risk was reduced by conducting some cognitive walkthroughs, allowing the applications to become more user friendly. There were also user interface changes that had to be made due to the software that was used to develop them. An example of this is the desktop application, the use of Java Swing GUI has reduced the usability as it is very limited.

Another risk raised in the risk analysis documentation is late changes to requirements. The agile approach has been used throughout the project, but due to development being slow at the start the backlog built up. This in return made us reconsider our core functionality and to make the applications as simple as we could. These late changes did affect what was completed but it made sure that there was some functionality in the applications.

In future, more preparation would be key and experience with the software before starting a project would be considerably beneficial as it would reduce the time spent on researching and learning. In terms of the user interface more time should have been allocated to usability testing to ensure that the interface is user friendly.

## Difficulties during the project

There were difficulties that had occurred due to lack of knowledge, this was overcome by looking at online resource and learning on the go. There was difficulty surrounding the usability testing. This was due to having to organise with other teams who had different availability time than our working schedule. Because of this we were not able to do as much usability testing as we initially thought we would be able to do. When nearing the fourth sprint there was a difficulty managing the expectations of the group, a large amount of work had been done but members felt like it wasn’t reflected in what we had produced so far.

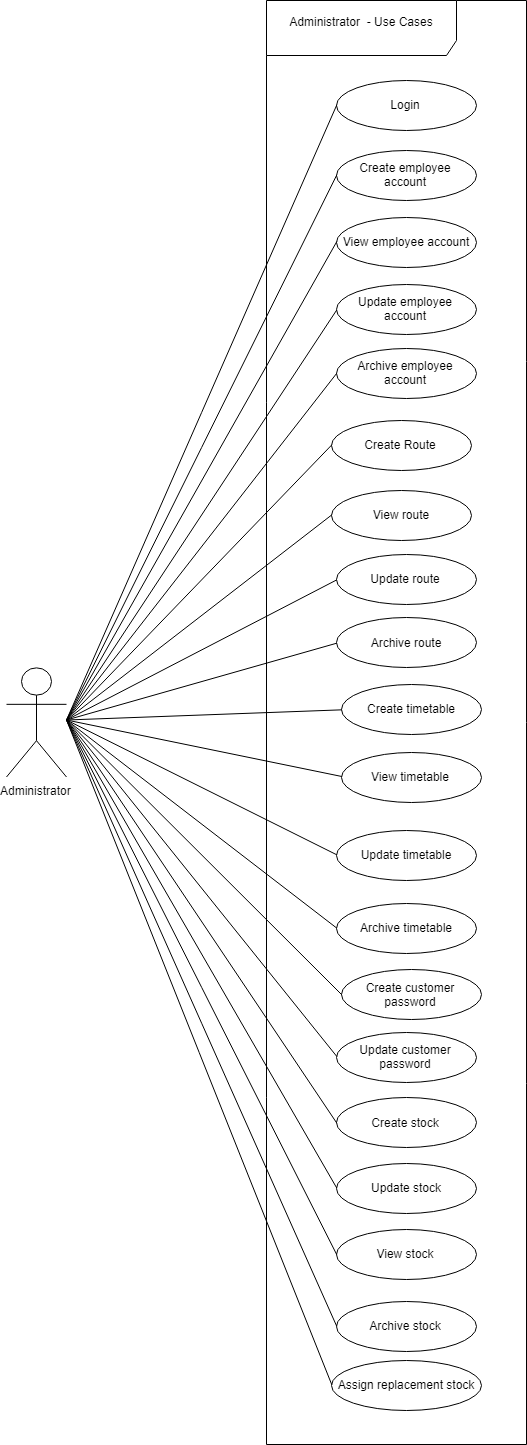
# Requirements Analysis and Design

## Product Backlog

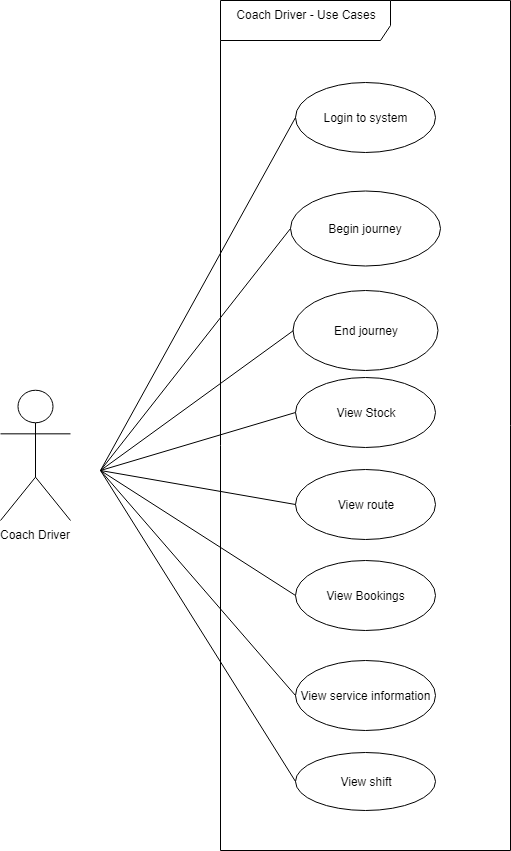
|  |  |  |
| --- | --- | --- |
| Backlog | In Progress | Complete |
| [DRIVER - DESKTOP] As a driver, I want to be able to accept an e-ticket booking reference from a passenger, to validate their journey on the coach. | [CUSTOMER - MOBILE] As a customer, I want to be able to view all my currently booked tickets and be able to display my booking reference code to the driver upon coach entry. | [CUSTOMER - MOBILE] As a customer, I want to be able to select the day and time in which I want to travel, so that available services during those times will be available for me to book. |
| [ADMIN - WEB] As an admin, I want to be able to approve customer account requests, so that customers can log in to their accounts. | [ADMIN - WEB] As an admin, I want to be able to create routes, so that coaches have a set path. | [CUSTOMER - MOBILE] As a customer, I want to be able to view my account details, so that I can check my information is correct. |
| [MANAGER - WEB] As a manager, I want to assign a driver a route for their shift, so that the driver can take passengers to their destinations. | [ADMIN - WEB] As an admin, I want to be able to update routes, in case a route needs to change. | [CUSTOMER - MOBILE] As a customer, I want to be able to create an account, so that I can book a place on a coach. |
| [MANAGER - WEB] As a manager, I want to be able to update and make changes to a driver's shift | [DRIVER - DESKTOP] As a driver, I want to be able to view coach capacity, so that the maximum number of passengers can be ascertained. | [CUSTOMER - MOBILE] As a customer, I want to be able to login to my account, so that I can access my account. |
| [CUSTOMER - MOBILE] As a customer, I want to be able to request the termination of my account, so that I can stop using the service. | [ADMIN - WEB] As an admin, I want to update journeys, so that I can view the routes and times. | [MANAGER - WEB] CHANGED TO ADMIN As a manager, I want to be able to view the driver's, so that I can see the drivers that are available. |
| [CUSTOMER - MOBILE] As a customer, I want to be able to know my ending station, so I can know where to get off the coach. | [ADMIN - WEB] As an admin, I want to archive journeys, so we know when journeys have been completed. | [MANAGER - WEB] CHANGED TO ADMIN As a manager, I want to view all available routes, so that I can see what routes will need covering. |
| [CUSTOMER - MOBILE] As a customer, searching for routes, I want to see fares and associated times for a journey, so that I can make sure I book the right journey for me. | [CUSTOMER - MOBILE] As a customer, I want to be able to view my prior travel history, so I can easily track where I have been and make new journey’s based on prior ones. | [MANAGER - WEB] CHANGED TO ADMIN As a manager, I want to view the journeys, so that I can see what time the routes are taken. |
| [DRIVER - DESKTOP] As a driver, I want to be able to view remaining seats, so that the amount of spare room can be ascertained. | [DRIVER - DESKTOP] As a driver, I want to be able to access the shift schedule, so that I know when I need to work. | [MANAGER - WEB] CHANGED TO ADMIN  As a manager, I want to be able to remove an employee’s shift, so that they no longer undertake that shift. |
| [DRIVER - DESKTOP] As a driver, I want to be able to view current stock, so that the number of coaches available to drive in the depot is known. | [ADMIN - WEB] As an admin, I want to be able to deploy replacement services in the event of a coach breakdown, so we can assist passengers. | [ADMIN - WEB] As an admin, I want to be able to retrieve routes, so that they can be decommissioned. |
| [DRIVER - DESKTOP] As a driver, I want to be able to view the set route for the journey, so that the coach can arrive at the destination. | [DRIVER - DESKTOP] As a driver, I want to be able to notify of breakdowns, so that a replacement coach will be called to assist passengers. | [ADMIN - WEB] As an admin, I want to retrieve the journeys, so that I can view the routes and times. |
| [CUSTOMER - MOBILE] As a customer, I want to be able to search for journeys given the starting point and destination. | [CUSTOMER - MOBILE] As a customer, I want to be able to make multiple bookings, so that I can travel with others. | [ADMIN - WEB] As an admin, I want to create journeys, so that the routes and times can be documented. |
| [CUSTOMER - MOBILE] As a customer, I want to be able to update my account details, so that my information can be up to date. | [CUSTOMER - MOBILE] As a customer, I want to be able to know the estimated time of the journey, so I know when the coach will get to my destination. | [ADMIN - WEB] As an admin, I want to be able to archive coaches in the stock inventory, so they can be decommissioned if not in use or replaced. |
| [ADMIN - WEB] As an admin, I want to be able to allow the changing of customer passwords, so that customers may alter their password. | [DRIVER - DESKTOP] As a driver, I want to be able to see the bookings made for a journey, so I know how many people will board the coach. | [ADMIN - WEB] As an admin, I want to be able to view coaches in the stock inventory, so that I can ascertain what coaches are currently available. |
|  | [DRIVER - DESKTOP] As a driver, I want to be able to stop a service for a valid route, so that I can provide the travel product for customers. | [ADMIN - WEB] As an admin, I want to able to update coaches, so that the status can be checked, and stock status can be updated. |
|  | [DRIVER - DESKTOP] As a driver, I want to be able to start a service for a valid route, so that I can provide the travel product for customers. | [ADMIN - WEB] As an admin, I want to be able to update employee accounts, so that information related to employees can be changed. |
|  | [CUSTOMER - MOBILE] As a customer, I want to be able to know my starting station, so I can know where to board the coach. | [ADMIN - WEB] As an admin, I want to be able to create records of new coaches, so that the stock can be documented. |
|  |  | [ADMIN - WEB] As an admin, I want to be able to login to the system, so that I can perform my responsibilities within the system. |
|  |  | [ADMIN - WEB] As an admin, I want to be able to retrieve employee accounts, so I can view information on employees. |
|  |  | [ADMIN - WEB] As an admin, I want to be able to archive employee accounts, so that they are no longer active when an employee leaves the company. |
|  |  | [CUSTOMER - MOBILE] As a customer, I want to be able to create bookings, so that I can catch the coach to my desired destination. |
|  |  | [DRIVER - DESKTOP] As a driver, I want to be able to log in to the system, so that the service can start. |
|  |  | [CUSTOMER - MOBILE] As a customer, I want to be able to pay via PayPal, so I can confirm my travel. |
|  |  | [MANAGER - WEB] CHANGED TO ADMIN As a manager, I want to be able to view the driver's, so that I can see the drivers that are available. |
|  |  | [ADMIN - WEB] As an admin, I want to be able to archive coaches in the stock inventory, so they can be decommissioned if not in use or replaced. |
|  |  | [MANAGER - WEB] As a manager, I want to be able to assign a driver a coach for the shift, so that the driver can perform his duties. |

## Use Case UML Diagrams

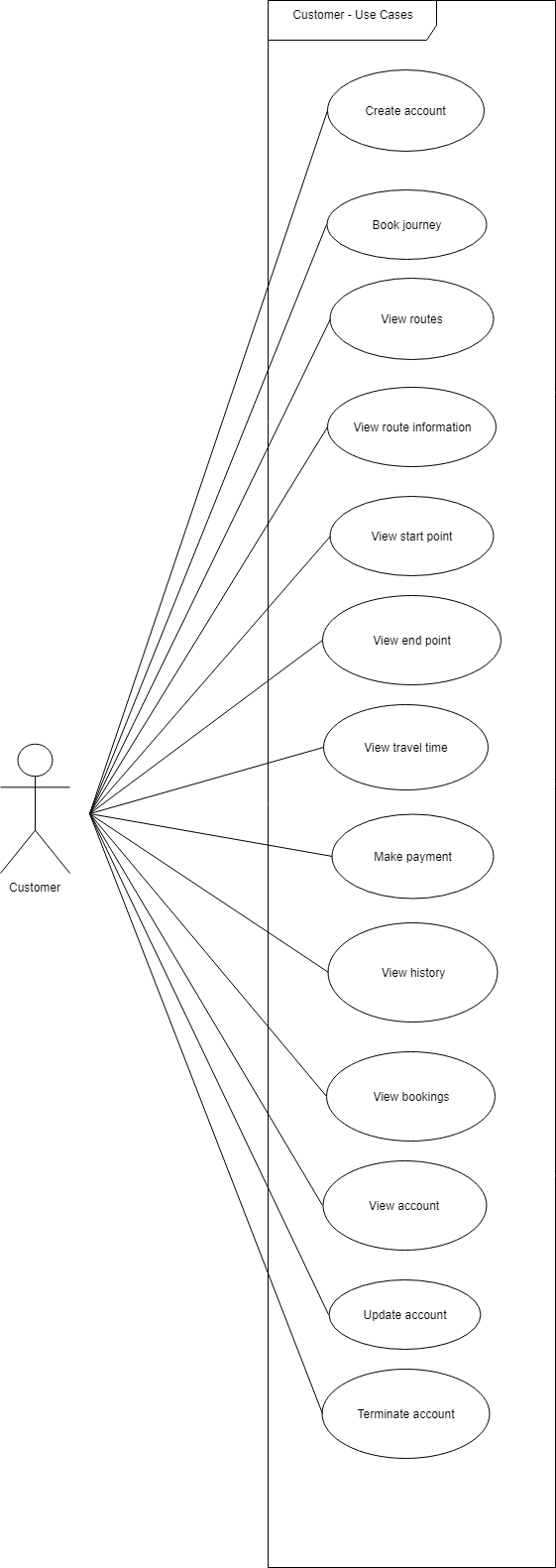
### Administrator



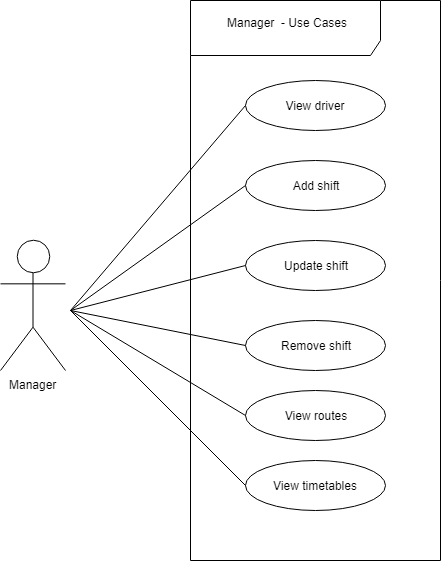
### Driver



### Customer



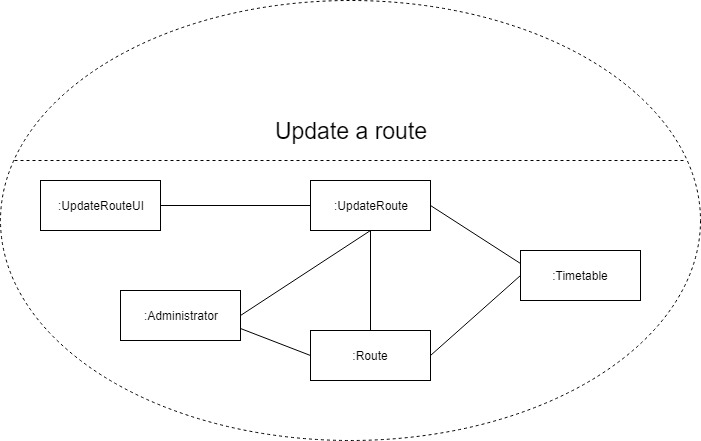
### Manager

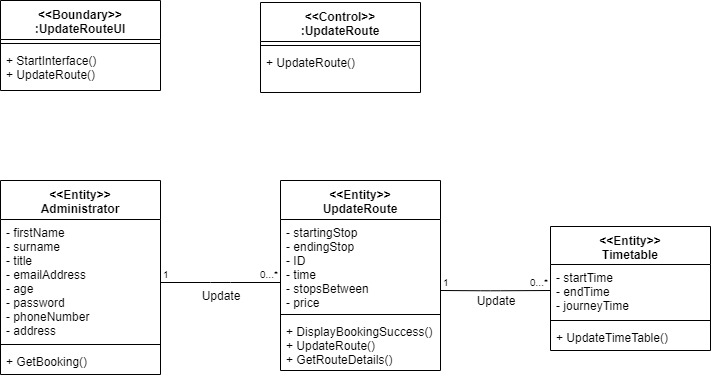
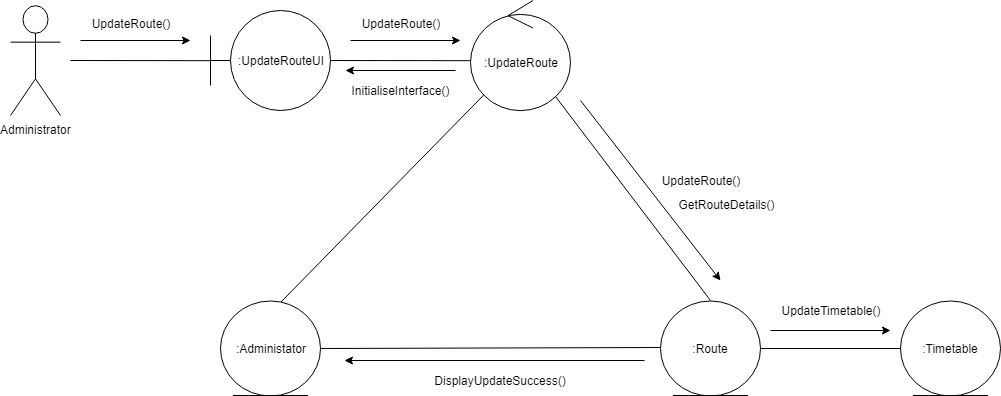


## Use Case Realisation

### Updating a Route



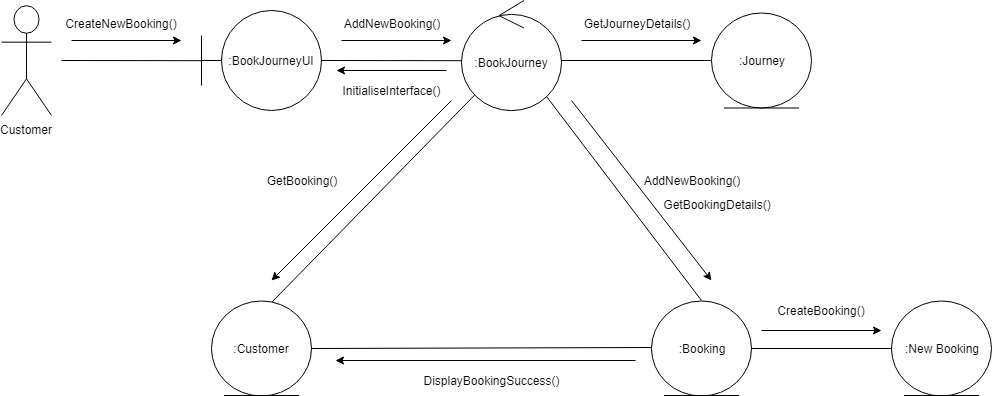
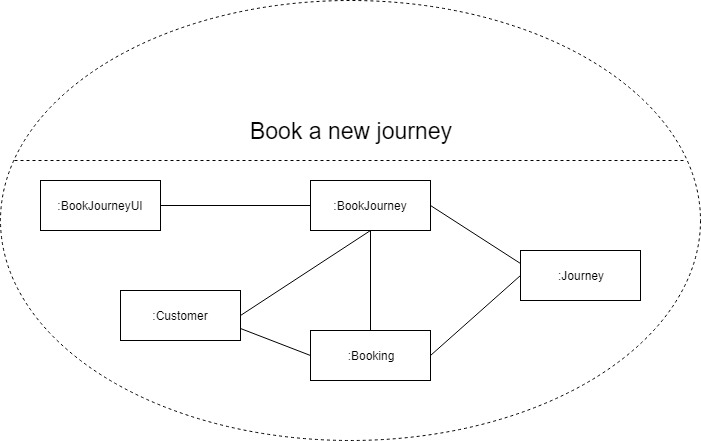


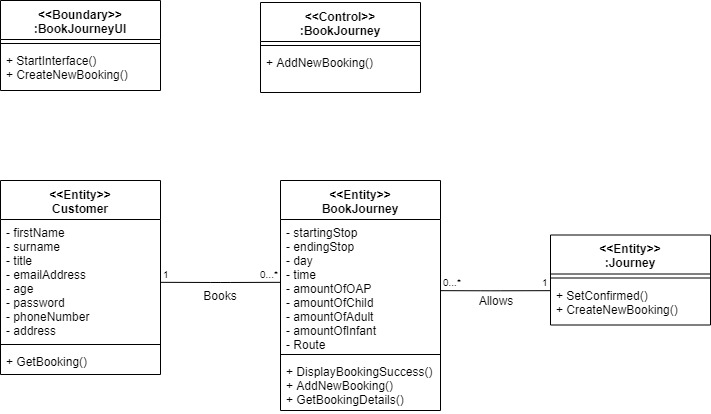


The administrator has the responsibility of updating the routes on the website application. To update the routes the administrator interacts with the user interface, enters the new data and submits. The API then receives a PUT request and processes it, this is then sent to the database to be stored in the route and timetable tables as these are the tables that will have the alterations.

### Booking onto a Journey







The customer has the responsibility of booking a journey on the mobile application. To do this they interact with the user interface by finding a journey, entering the number of passengers and pressing book. When finding a journey, the API uses a GET request to get the specified information from the journeys table in the database. The final action of pressing book triggers an API POST request. This request sends the new booking to the database to store in the booking table.

## Sprint Plan

### Sprint 1

**User Stories addressed by William Butler**

Mobile App - As a customer, I want to be able to login to my account, so that I can access my account. - Completed

Web App - As an admin, I want to be able to retrieve employee accounts, so I can view information on employees. - Completed

Web App - As an admin, I want to be able to archive employee accounts, so that they are no longer active when an employee leaves the company. - Completed

**User Stories addressed by Goel Biju**

Mobile App - As a customer, I want to be able to create an account, so that I can book a place on a coach. - Completed

Web App - As an admin, I want to be able to create employee accounts, so that new employees can undertake their duties. - Completed

Desktop App - As a driver, I want to be able to access the shift schedule, so that I know when I need to work. - Completed

**User Stories addressed by Andrew Bellas**

Desktop App - As a driver, I want to be able to log in to the system, so that the service can start. - Completed

Web App - As an admin, I want to be able to log in to the system, so that I can perform my responsibilities within the system. - Completed

Mobile App - As a customer, I want to be able to view my account details, so that I can check my information is correct. - Completed

**User Stories addressed by Vincent Castellani**

Desktop App - As a driver, I want to be able to view current stock, so that the number of coaches available to drive in the depot is known. - Completed

Mobile App - As a customer, I want to be able to update my account details, so that my information can be up to date.

Web App - As an admin, I want to be able to update employee accounts, so that information related to employees can be changed. – Completed

### Sprint 2

**User Stories addressed by William Butler**

Desktop App - As a driver, I want to be able to see the bookings made for a journey, so I know how many people will board the coach.

Mobile App - As a customer, I want to be able to select the day and time in which I want to travel, so that available services during those times will be available for me to book. – Completed

Web App - As an admin, I want to be able to create records of new coaches, so that the stock can be documented. – Completed

**User Stories addressed by Goel Biju**

Web App - As an admin, I want to be able to archive coaches in the stock inventory, so they can be decommissioned if not in use or replaced. – Completed

Web App - As an admin, I want to be able to view driver's journeys and the services they have performed, so each driver's actions can be monitored.

Desktop App - As a driver, I want to be able to stop a service for a valid route, so that I can provide the travel product for customers.

**User Stories addressed by Andrew Bellas**

Desktop App - As a driver, I want to be able to start a service for a valid route, so that I can provide the travel product for customers.

Web App - As an admin, I want to able to update coaches, so that the status can be checked, and stock status can be updated. – Completed

Mobile App - As a customer, I want to be able to know the estimated time of the journey, so I know when the coach will get to my destination.

**User Stories addressed by Vincent Castellani**

Web App - As an admin, I want to be able to allow the changing of customer passwords, so that customers may alter their password. \*\*Low Priority\*\*

Mobile App - As a customer, I want to be able to search for timetabled routes given the starting point and destination.

Web App - As an admin, I want to be able to view coaches in the stock inventory, so that I can ascertain what coaches are currently available. – Completed

### Sprint 3

**User Stories addressed by William Butler**

Desktop App - As a driver, I want to be able to accept an e-ticket booking reference from a passenger, to validate their journey on the coach.

Mobile App - As a customer, I want to be able to pay via PayPal, so I can confirm my travel. – Completed

Web App - As an admin, I want to create journeys, so that the routes and times can be documented. – Completed

Web App - As an admin, I want to retrieve the journeys, so that I can view the routes and times. – Completed

**User Stories addressed by Goel Biju**

Mobile App - As a customer. I want to be able to make multiple bookings, so that I can travel with others. – Completed

Web App - As an admin, I want to be able to archive routes, so that they can be decommissioned.

Web App\*\* - As an admin, I want to update journeys, so that the routes and times can be amended. \*\*CHANGE\*\*

Web App - As a manager, I want to be able to assign a driver a coach for the shift, so that the driver can perform his duties. - Adding a new shift for a driver. – Completed

**User Stories addressed by Andrew Bellas**

Desktop App - As a driver, I want to be able to view coach capacity, so that the maximum number of passengers can be ascertained. – Completed

Web App - As an admin, I want to be able to retrieve routes, so that I can view them. – Completed

Mobile App - As a customer, I want to be able to view all my currently booked tickets and be able to display my booking reference code to the driver upon coach entry. – Completed

**User Stories addressed by Vincent Castellani**

Desktop App - As a driver, I want to be able to notify of breakdowns, so that a replacement coach will be called to assist passengers.

Mobile App - As a customer, I want to be able to create bookings, so that I can catch the coach to my desired destination. – Completed

Web App - As a manager, I want to be able to view the drivers, so that I can see the driver’s that are available. – Completed

### Sprint 4

**User Stories addressed by William Butler**

Web App - As a manager, I want to be able to update and make changes to a driver shift.

Mobile App - As a customer, I want to be able to view my prior travel history, so I can easily track where I have been and make new journey’s based on prior ones.

Web App - As a manager, I want to view all available routes, so that I can see what routes will need covering.

**User Stories addressed by Goel Biju**

Desktop App - As a driver, I want to be able to view the set route for the journey, so that the coach can arrive at the destination.

Mobile App - As a customer, searching for routes, I want to see fares and associated times for a journey, so that I can make sure I book the right journey for me.

Web App - As an admin, I want to be able to deploy replacement services in the event of a coach breakdown, so we can assist passengers.

Desktop App - As a driver, I want to be able to accept a e-ticket booking reference from a passenger, to validate their journey on the coach.

**User Stories addressed by Andrew Bellas**

Web App - As an admin, I want to be able to deploy replacement services in the event of a coach breakdown, so we can assist passengers.

Mobile App - As a customer, I want to be able to know my starting station, so I can know where to board the coach. – Completed

Mobile App - As a customer, I want to be able to know my ending station, so I can know where to get off the coach. – Completed

**User Stories addressed by Vincent Castellani**

Desktop App - As a driver, I want to be able to view remaining seats, so that the amount of spare room can be ascertained.

Web App - As a manager, I want to view the journeys, so that I can see what time the routes are taken. - Already been done as another user story was the same thing for the admin. – Completed

Web App - As a manager, I want to be able to remove an employee’s shift, so that they no longer undertake that shift. – Completed

## Changes During Development

At the start of our project the initial requirements were decided, from these requirements the minimum viable product was decided. The requirements and minimum viable product were reassessed at the end of sprint 3 so that the basic functionality could be completed and functioning.

The user story – as a driver, I want to be able to view current stock. This was removed because it isn’t something the driver would need to do as they would be already assigned a coach when they have logged in. In the initial requirements it was decided that we would have an administrator, manager, driver and customer.

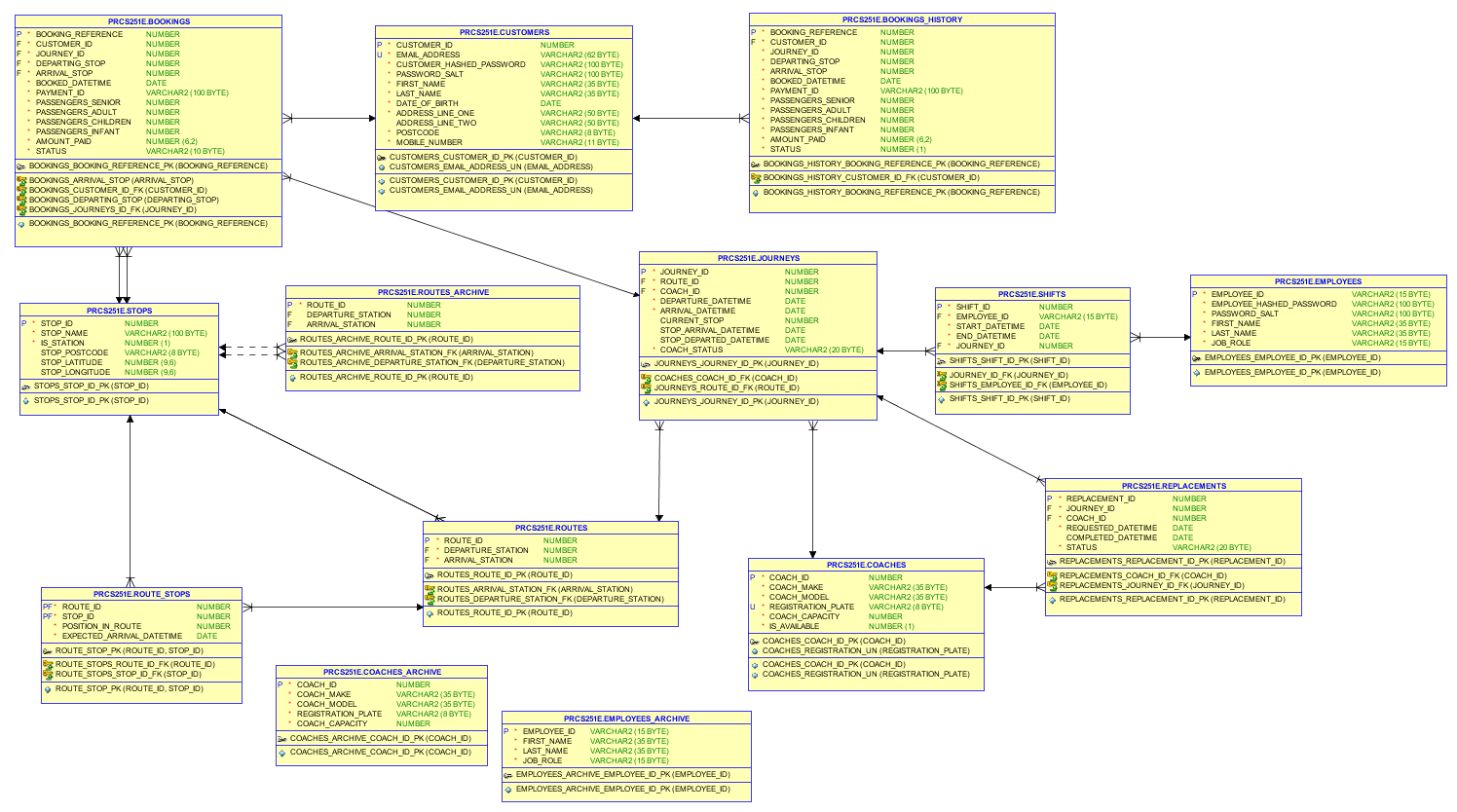
When moving onto the manager user stories, due to the web login system not being able to distinguish between users it was decided that the manager would be removed. All manager user stories were changed to administrator.

The database has been altered several times to accommodate functionality changes. The initial idea of having timetables was changed to journeys which incorporates the timetable attributes. This change was made to improve data integrity and to improve the connection between what route and coaches would be used for each journey. This then makes the connection between the shifts and journey stronger and when a shift is assigned an employee all the correct information is retrieved.

During the fourth sprint the core functionality was decided, this functionality was prioritised to attempt to get a system that has the basics working. The user stories were marked with core and were to be completed before any other user story.

# Database Design

## Database Diagram



## SQL Create Statements

### Employee Table

Table

This table stores data for employees so that they can log in and use the system and also be assigned to shifts:

CREATE TABLE employees(

employee\_id VARCHAR2(15)

CONSTRAINT employees\_employee\_id\_pk PRIMARY KEY,

employee\_hashed\_password VARCHAR2(100)

CONSTRAINT employees\_employee\_password\_nn NOT NULL,

-- CONSTRAINT employees\_employee\_password\_chk

-- CHECK (REGEXP\_LIKE(employee\_password, '^[a-z0-9A-Z]{7,16}$')),

password\_salt VARCHAR2(100)

CONSTRAINT employees\_password\_salt\_nn NOT NULL,

first\_name VARCHAR2(35)

CONSTRAINT employees\_first\_name\_nn NOT NULL

CONSTRAINT employees\_first\_name\_chk\_alpha

CHECK (REGEXP\_LIKE(first\_name, '^[A-Za-z-'']+$'))

CONSTRAINT employees\_first\_name\_chk\_initcap

CHECK (first\_name = INITCAP(first\_name)),

last\_name VARCHAR2(35)

CONSTRAINT employees\_last\_name NOT NULL

CONSTRAINT employees\_last\_name\_chk\_alpha

CHECK (REGEXP\_LIKE(last\_name, '^[A-Za-z-'']+$'))

CONSTRAINT employees\_last\_name\_chk\_initcap

CHECK (last\_name = INITCAP(last\_name)),

job\_role VARCHAR2(15)

CONSTRAINT employees\_job\_role NOT NULL

CONSTRAINT employees\_job\_role\_chk

CHECK (job\_role IN ('Driver', 'Administrator', 'Manager'))

);

Sequence

This sequence handles the employee ID number to be assigned to an employee:

CREATE SEQUENCE seq\_employee\_id

START WITH 1000

INCREMENT BY 1

NOCYCLE

NOCACHE;

Triggers

This trigger is used to initiate an employee account. It determines what role an employee is and add the prefix to the employee ID along with the sequence value. If the role is not recognised it will throw an error.

CREATE OR REPLACE TRIGGER trg\_employee\_initalise

BEFORE INSERT ON employees FOR EACH ROW

BEGIN

/\* Check what the job role is and assign the correct employee prefix

Administrator/Driver (A/D) \*/

IF :NEW.job\_role = 'Driver' THEN

:NEW.employee\_id := 'D' || seq\_employee\_id.nextval;

ELSEIF :NEW.job\_role = 'Administrator' THEN

:NEW.employee\_id := 'A' || seq\_employee\_id.nextval;

ELSE

RAISE\_APPLICATION\_ERROR(-2000, 'Could not recognise the job role.');

END IF;

END;

### Employees Archive

Table

This table is the archive for employee details when an employee leaves the company:

CREATE TABLE employees\_archive(

employee\_id VARCHAR2(15)

CONSTRAINT employees\_archive\_employee\_id\_pk PRIMARY KEY,

first\_name VARCHAR2(35)

CONSTRAINT employees\_archive\_first\_name NOT NULL,

--CONSTRAINT employees\_archive\_first\_name\_chk\_alpha

-- CHECK (REGEXP\_LIKE(first\_name, '^[A-Za-z-'']+$'))

--CONSTRAINT employees\_archive\_first\_name\_chk\_initcap

-- CHECK (first\_name = INITCAP(first\_name)),

last\_name VARCHAR2(35)

CONSTRAINT employees\_archive\_last\_name NOT NULL,

--CONSTRAINT employees\_archive\_last\_name\_chk\_alpha

-- CHECK (REGEXP\_LIKE(last\_name, '^[A-Za-z-'']+$'))

--CONSTRAINT employees\_archive\_last\_name\_chk\_initcap

-- CHECK (last\_name = INITCAP(last\_name)),

job\_role VARCHAR2(15)

CONSTRAINT employees\_archive\_job\_role NOT NULL

--CONSTRAINT employees\_archive\_job\_role\_chk

-- CHECK (job\_role IN ('Driver', 'Administrator', 'Manager'))

);

Triggers

Trigger to archive employee accounts when deleted from the main table:

CREATE OR REPLACE TRIGGER trg\_archive\_employees

BEFORE DELETE ON employees FOR EACH ROW

BEGIN

INSERT INTO employees\_archive

(employee\_id, first\_name, last\_name, job\_role)

VALUES

(:OLD.employee\_id, :OLD.first\_name, :OLD.last\_name, :OLD.job\_role);

END;

### Shifts

Table

This table stores shifts assigned to drivers so that drivers can see when they are working:

CREATE TABLE shifts(

shift\_id NUMBER

CONSTRAINT shifts\_shift\_id\_pk PRIMARY KEY,

employee\_id VARCHAR2(15) NOT NULL

CONSTRAINT shifts\_employee\_id\_fk REFERENCES employees(employee\_id),

start\_datetime DATE

CONSTRAINT shifts\_start\_datetime\_nn NOT NULL,

end\_datetime DATE

CONSTRAINT shifts\_end\_datetime\_nn NOT NULL

);

Sequence

This sequence handles the shift ID as the primary key for the table:

CREATE SEQUENCE seq\_shift\_id

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

Triggers

This trigger increments the above sequence when a new shift is created:

CREATE OR REPLACE TRIGGER trg\_shift\_initialise

BEFORE INSERT ON shifts FOR EACH ROW

BEGIN

SELECT seq\_shift\_id.nextval

INTO :NEW.shift\_id

FROM sys.dual;

END;

### Customers

Table

This table stores the details of our customers so that they can log into the system and make and get bookings for coach journeys:

CREATE TABLE customers(

customer\_id NUMBER

CONSTRAINT customers\_customer\_id\_pk PRIMARY KEY,

email\_address VARCHAR2(62)

CONSTRAINT customers\_email\_address\_un UNIQUE

CONSTRAINT customers\_email\_address\_nn NOT NULL

CONSTRAINT customers\_email\_address\_chk

CHECK (REGEXP\_LIKE(email\_address, '^\w+(\.\w+)\*@\w+(\.\w+)+$')),

customer\_hashed\_password VARCHAR2(100)

CONSTRAINT customers\_customer\_hashed\_password\_nn NOT NULL,

-- CONSTRAINT customers\_customer\_password\_chk

-- CHECK (REGEXP\_LIKE(customer\_password, '^[a-z0-9A-Z]{7,16}$')),

password\_salt VARCHAR2(100)

CONSTRAINT customers\_password\_salt\_nn NOT NULL,

first\_name VARCHAR2(35)

CONSTRAINT customers\_first\_name\_nn NOT NULL

CONSTRAINT customers\_first\_name\_chk\_alpha

CHECK (REGEXP\_LIKE(first\_name, '^[A-Za-z-'']+$'))

CONSTRAINT customers\_first\_name\_chk\_initcap

CHECK (first\_name = INITCAP(first\_name)),

last\_name VARCHAR2(35)

CONSTRAINT customers\_last\_name\_nn NOT NULL

CONSTRAINT customers\_last\_name\_chk\_alpha

CHECK (REGEXP\_LIKE(last\_name, '^[A-Za-z-'']+$'))

CONSTRAINT customers\_last\_name\_chk\_initcap

CHECK (last\_name = INITCAP(last\_name)),

date\_of\_birth DATE

CONSTRAINT customers\_date\_of\_birth\_nn NOT NULL,

address\_line\_one VARCHAR2(50)

CONSTRAINT customers\_address\_line\_one\_nn NOT NULL

CONSTRAINT customers\_address\_line\_one\_chk

CHECK (REGEXP\_LIKE(address\_line\_one, '^[A-Za-z0-9 -]+$')),

address\_line\_two VARCHAR2(50)

CONSTRAINT customers\_address\_line\_two\_chk

CHECK (REGEXP\_LIKE(address\_line\_two, '^[A-Za-z0-9 -]+$')),

postcode VARCHAR2(8)

CONSTRAINT customers\_postcode\_nn NOT NULL

CONSTRAINT customers\_postcode\_chk

CHECK (REGEXP\_LIKE(postcode, '^[A-Z0-9 ]+$')),

mobile\_number VARCHAR2(11)

CONSTRAINT customers\_mobile\_number\_nn NOT NULL

CONSTRAINT customers\_mobile\_number\_chk

CHECK (REGEXP\_LIKE(mobile\_number, '^07[0-9]{9}$'))

);

Sequence

This sequence handles the customer ID so that it is unique for the primary key of the customer table:

CREATE SEQUENCE seq\_customer\_id

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

Triggers

This trigger is used to increment the sequence above when a new customer is created:

CREATE OR REPLACE TRIGGER trg\_customer\_initalise

BEFORE INSERT ON customers FOR EACH ROW

BEGIN

SELECT seq\_customer\_id.nextval

INTO :NEW.customer\_id

FROM sys.dual;

END;

### Stops

Table

This table handles all of the stops the company currently serves with our routes and stores the locations and names of them:

CREATE TABLE stops(

stop\_id NUMBER

CONSTRAINT stops\_stop\_id\_pk PRIMARY KEY,

stop\_name VARCHAR2(100)

CONSTRAINT stops\_stop\_name\_nn NOT NULL,

is\_station NUMBER(1)

CONSTRAINT stops\_is\_station\_nn NOT NULL

CONSTRAINT is\_station\_check

CHECK (is\_station IN (0, 1)),

stop\_postcode VARCHAR2(8),

stop\_latitude NUMBER(9, 6),

stop\_longitude NUMBER(9, 6)

);

Sequence

This sequence handles the stop ID so that it is unique for the primary key of the stop table:

CREATE SEQUENCE seq\_stop\_id

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

Triggers

This trigger initiate the stop id when a new stop is added based off of the sequence above:

CREATE OR REPLACE TRIGGER trg\_stop\_initalise

BEFORE INSERT ON stops FOR EACH ROW

BEGIN

SELECT seq\_stop\_id.nextval

INTO :NEW.stop\_id

FROM sys.dual;

END;

### Routes

Table

This table contains the routes that coaches take with the starting and ending stops for that route:

CREATE TABLE routes(

route\_id NUMBER

CONSTRAINT routes\_route\_id\_pk PRIMARY KEY,

departure\_station NUMBER NOT NULL

CONSTRAINT routes\_departure\_station\_fk REFERENCES stops (stop\_id),

arrival\_station NUMBER NOT NULL

CONSTRAINT routes\_arrival\_station\_fk REFERENCES stops (stop\_id)

);

Sequence

This sequence is for managing the route ID for the table:

CREATE SEQUENCE seq\_route\_id

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

Triggers

This trigger initiates the route ID when new records are added to the table:

CREATE OR REPLACE TRIGGER trg\_route\_initialise

BEFORE INSERT ON routes FOR EACH ROW

BEGIN

SELECT seq\_route\_id.nextval

INTO :NEW.route\_id

FROM sys.dual;

END;

### Routes Archive

Table

This table is the archive for routes so that routes aren’t permanently lost when decommissioned and can be reused in the future:

CREATE TABLE routes\_archive(

route\_id NUMBER

CONSTRAINT routes\_archive\_route\_id\_pk PRIMARY KEY,

departure\_station NUMBER

CONSTRAINT routes\_archive\_departure\_station\_fk REFERENCES stops (stop\_id),

arrival\_station NUMBER

CONSTRAINT routes\_archive\_arrival\_station\_fk REFERENCES stops (stop\_id)

);

### Route Stops

Table

This is a link table to assign stops to a route so that it can be seen where a route stops at on its journey:

CREATE TABLE route\_stops(

route\_id NUMBER NOT NULL

CONSTRAINT route\_stops\_route\_id\_fk REFERENCES routes (route\_id),

stop\_id NUMBER NOT NULL

CONSTRAINT route\_stops\_stop\_id\_fk REFERENCES stops (stop\_id),

position\_in\_route NUMBER

CONSTRAINT position\_in\_route\_nn NOT NULL,

CONSTRAINT route\_stop\_pk PRIMARY KEY (route\_id, stop\_id)

);

### Coaches

Table

The coaches table stores the details of all the coaches currently in the fleet to be used by drivers on journeys:

CREATE TABLE coaches(

coach\_id NUMBER

CONSTRAINT coaches\_coach\_id\_pk PRIMARY KEY,

coach\_make VARCHAR2(35)

CONSTRAINT coaches\_coach\_make\_nn NOT NULL,

coach\_model VARCHAR2(35)

CONSTRAINT coaches\_coach\_model\_nn NOT NULL,

registration\_plate VARCHAR2(8)

CONSTRAINT coaches\_registration\_un UNIQUE

CONSTRAINT coaches\_registration\_nn NOT NULL,

coach\_capacity NUMBER

CONSTRAINT coaches\_coach\_capacity\_nn NOT NULL,

is\_available NUMBER(1)

CONSTRAINT coaches\_is\_available\_nn NOT NULL

CONSTRAINT is\_available\_chk

CHECK (is\_available IN (0, 1))

);

Sequence

The sequence gives entries in the database a unique number as their primary key:

CREATE SEQUENCE seq\_coach\_id

START WITH 1

INCREMENT BY 1

MAXVALUE 10000

CYCLE

NOCACHE;

Triggers

This trigger uses the sequence above to assign it to new incoming records:

CREATE OR REPLACE TRIGGER TRG\_COACH\_INITIALISE

BEFORE INSERT ON coaches FOR EACH ROW

BEGIN

SELECT seq\_coach\_id.nextval

INTO :NEW.coach\_id

FROM sys.dual;

END;

### Coaches Archive

Table

This table archives any coaches that have been decommissioned or sold so that they are still on record for any administrative needs:

CREATE TABLE coaches\_archive(

coach\_id NUMBER

CONSTRAINT coaches\_archive\_coach\_id\_pk PRIMARY KEY,

coach\_make VARCHAR2(35)

CONSTRAINT coaches\_archive\_coach\_make\_nn NOT NULL,

coach\_model VARCHAR2(35)

CONSTRAINT coaches\_archive\_coach\_model\_nn NOT NULL,

registration\_plate VARCHAR2(8)

CONSTRAINT coaches\_archive\_registration\_plate\_nn NOT NULL,

coach\_capacity NUMBER

CONSTRAINT coaches\_archive\_coach\_capacity\_nn NOT NULL

);

Triggers

This trigger sends data from the main coaches table when it is due to be deleted:

CREATE OR REPLACE TRIGGER trg\_archive\_coaches

BEFORE DELETE ON coaches FOR EACH ROW

BEGIN

INSERT INTO coaches\_archive

(coach\_id, coach\_make, coach\_model, registration\_plate, coach\_capacity)

VALUES

(:OLD.coach\_id, :OLD.coach\_make, :OLD.coach\_model, :OLD.registration\_plate,

:OLD.coach\_capacity);

END;

### Journeys

Table

The journey table assigns a route with a shift, coach and employee so that a trip can be carried out and accept bookings:

CREATE TABLE journeys(

journey\_id NUMBER NOT NULL

CONSTRAINT journeys\_journey\_id\_pk PRIMARY KEY,

route\_id NUMBER NOT NULL

CONSTRAINT journeys\_route\_id\_fk REFERENCES routes(route\_id),

shift\_id NUMBER NOT NULL

CONSTRAINT journeys\_shift\_id\_fk REFERENCES shifts(shift\_id),

coach\_id NUMBER NOT NULL

CONSTRAINT coaches\_coach\_id\_fk REFERENCES coaches(coach\_id),

--employee\_id VARCHAR(15) NOT NULL

-- CONSTRAINT employees\_employee\_id\_fk REFERENCES employees (employee\_id),

departure\_datetime DATE

CONSTRAINT journeys\_departure\_datetime\_nn NOT NULL,

arrival\_datetime DATE

CONSTRAINT journeys\_arrival\_datetime\_nn NOT NULL,

current\_stop NUMBER,

stop\_arrival\_datetime DATE,

stop\_departed\_datetime DATE,

coach\_status VARCHAR2(20)

CONSTRAINT journeys\_coach\_status\_nn NOT NULL

CONSTRAINT journeys\_coach\_status\_chk

CHECK (coach\_status IN

('Scheduled', 'Departed', 'On-route', 'At Stop', 'Arrived', 'Broken Down', 'Replacement Deployed',

'Cancelled', 'Complete'))

);

Sequence

Initiates the journey ID so that it is unique:

CREATE SEQUENCE seq\_journey\_id

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

Triggers

Executes when a new journey is to be added to increment and assign the ID:

CREATE OR REPLACE TRIGGER trg\_journey\_initalise

BEFORE INSERT ON journeys FOR EACH ROW

BEGIN

SELECT seq\_journey\_id.nextval

INTO :NEW.journey\_id

FROM sys.dual;

END;

This trigger checks that the journey start date is in the future and throws an error if this is not the case:

CREATE OR REPLACE TRIGGER trg\_check\_journey\_start\_date

BEFORE INSERT ON journeys

FOR EACH ROW

BEGIN

IF( :new.departure\_datetime < sysdate)

THEN

RAISE\_APPLICATION\_ERROR(

-20001,

'Journeys need to made in future and not today' );

END IF;

END;

This trigger checks if the journey end date is in the future and after the departure time throws an error if this is not the case:

CREATE OR REPLACE TRIGGER trg\_check\_journey\_end\_date

BEFORE INSERT ON journeys

FOR EACH ROW

BEGIN

IF( (:new.arrival\_datetime < sysdate) AND (:new.arrival\_datetime > :new.departure\_datetime))

THEN

RAISE\_APPLICATION\_ERROR(

-20001,

'Journeys need to made in future and not today' );

END IF;

END;

### Bookings

Table

The booking table stores the booking details for a customer so that they can be verified by the driver on the day of travel:

CREATE TABLE bookings(

booking\_reference NUMBER NOT NULL

CONSTRAINT bookings\_booking\_reference\_pk PRIMARY KEY,

customer\_id NUMBER NOT NULL

CONSTRAINT bookings\_customer\_id\_fk REFERENCES customers (customer\_id),

journey\_id NUMBER NOT NULL

CONSTRAINT bookings\_journeys\_id\_fk REFERENCES journeys (journey\_id),

departing\_stop NUMBER NOT NULL

CONSTRAINT bookings\_departing\_stop REFERENCES stops (stop\_id),

arrival\_stop NUMBER NOT NULL

CONSTRAINT bookings\_arrival\_stop REFERENCES stops (stop\_id),

booked\_datetime DATE

CONSTRAINT bookings\_booked\_datetime NOT NULL,

--amount\_of\_people NUMBER,

passengers\_senior NUMBER

CONSTRAINT bookings\_passengers\_senior\_nn NOT NULL,

passengers\_adult NUMBER

CONSTRAINT bookings\_passengers\_adult\_nn NOT NULL,

passengers\_teenager NUMBER

CONSTRAINT bookings\_passengers\_teenager\_nn NOT NULL,

passengers\_infant NUMBER

CONSTRAINT bookings\_passengers\_infant\_nn NOT NULL,

amount\_paid NUMBER(6, 2)

CONSTRAINT bookings\_amount\_paid\_nn NOT NULL,

status VARCHAR2(10)

CONSTRAINT bookings\_status\_nn NOT NULL

CONSTRAINT bookings\_status\_chk

CHECK(status IN ('Confirmed', 'Checked-in', 'Complete'))

);

Sequence

This sequence initiates the booking reference which is used by the driver to verify right to travel:

CREATE SEQUENCE seq\_booking\_reference

START WITH 300

INCREMENT BY 1

NOCYCLE

NOCACHE;

Triggers

This trigger inserts the booking reference when a booking is completed:

CREATE OR REPLACE TRIGGER trg\_booking\_initialise

BEFORE INSERT ON bookings FOR EACH ROW

BEGIN

SELECT seq\_booking\_reference.nextval

INTO :NEW.booking\_reference

FROM sys.dual;

END;

### Booking History

Table

This table stores bookings once they have been completed so that they can be viewed by a customer to see their history:

CREATE TABLE bookings\_history(

booking\_reference NUMBER

CONSTRAINT bookings\_history\_booking\_reference\_pk PRIMARY KEY,

customer\_id NUMBER NOT NULL

CONSTRAINT bookings\_history\_customer\_id\_fk REFERENCES customers (customer\_id),

journey\_id NUMBER

CONSTRAINT bookings\_history\_journey\_id\_nn NOT NULL,

departing\_stop NUMBER

CONSTRAINT bookings\_history\_departing\_stop\_nn NOT NULL,

arrival\_stop NUMBER

CONSTRAINT bookings\_history\_arrival\_stop\_nn NOT NULL,

booked\_datetime DATE

CONSTRAINT bookings\_history\_booked\_datetime\_nn NOT NULL,

passengers\_senior NUMBER

CONSTRAINT bookings\_history\_passengers\_senior\_nn NOT NULL,

passengers\_adult NUMBER

CONSTRAINT bookings\_history\_passengers\_adult\_nn NOT NULL,

passengers\_teenager NUMBER

CONSTRAINT bookings\_history\_passengers\_teenager\_nn NOT NULL,

passengers\_infant NUMBER

CONSTRAINT bookings\_history\_passengers\_infant\_nn NOT NULL,

amount\_paid NUMBER(6, 2)

CONSTRAINT bookings\_history\_amount\_paid\_nn NOT NULL,

amount\_of\_people NUMBER

CONSTRAINT bookings\_history\_amount\_of\_people NOT NULL,

status NUMBER(1)

CONSTRAINT bookings\_history\_status NOT NULL

);

### Replacements

Table

This table stores replacement coach details so it is clear what coach and which driver has been assign to resolve the issue:

CREATE TABLE replacements(

replacement\_id NUMBER

CONSTRAINT replacements\_replacement\_id\_pk PRIMARY KEY,

journey\_id NUMBER NOT NULL

CONSTRAINT replacements\_journey\_id\_fk REFERENCES journeys (journey\_id),

coach\_id NUMBER NOT NULL

CONSTRAINT replacements\_coach\_id\_fk REFERENCES coaches (coach\_id),

shift\_id NUMBER NOT NULL

CONSTRAINT replacements\_shift\_id\_fk REFERENCES shifts (shift\_id),

requested\_datetime DATE

CONSTRAINT replacements\_request\_datetime\_nn NOT NULL,

completed\_datetime DATE,

status VARCHAR2(20)

CONSTRAINT replacements\_status\_nn NOT NULL

CONSTRAINT replacements\_status\_chk

CHECK (status IN

('Waiting', 'Deployed', 'Complete', 'Cancelled', 'Incomplete'))

);

## Normalised Tables

### First Normalised Tables

### Second Normalised Tables



The second normalisation changed the timetables to journeys and we also added employee archive as we thought it would be important to be able to save the employees details. The shifts had start and end of shifts attributes added for more functionality.

### Class Diagram Comparison to Normalised Tables

The initial normalisation had the attributes and tables that were needed for the base of the applications. Throughout each sprint the database was updated to match the user stories. An example of this are the tables that were added for bookings, the table was not needed until later in the project and was not considered at the start because we focused on other functionality.

The final database in comparison to the normalised tables is considerably different due to the changes in each sprint. Some of the tables had name changes, attributes added or taken away. The nature of these changes is mainly due to our agile approach and the “single-slice” of the database, middleware and client that group members worked on in order to complete their respectful user stories.

# Usability

## Cognitive Walkthrough Analysis

### Outline

In terms of cognitive walkthrough, an analysis was taken of all three applications, in order to assess the end user’s ease of use, and to facilitate the best user experience. Over the responses, data was ascertained, and formative feedback was considered.

The cognitive walkthroughs was conducted based on 8 questions, designed to ascertain experience with the app, the ease of completing tasks set, features which were perceived as difficult to use, any features that felt missing, favourite features, least favourite features, and how in control a person felt while using the application wireframes (relating to Schneiderman’s rule of supporting a user’s internal locus of control).

### User evaluation feedback priority list (in order of feedback received and changes needed, gained from feedback)

#### Website app

In terms of the website cognitive walkthrough, the positive comments made about the ease of use of the site are that it feels fluid, feels feature complete usage-wise, and that the side menu was, for the most part, easy to understand. However, in terms of comments received that were negative, the first was a confusion in navigating certain elements of the site wireframes, in terms of the tables on certain pages, used to show data about, for example, the viewing of employee accounts, which was titled ‘retrieve employees’. This was later corrected, and other such navigation items, to terminology such as ‘View employees’, based on this feedback and similar feedback regarding how the side menu of the site could be grouped more appropriately with more consistent naming conventions.

Another criticism was the difficulty in understanding data when viewing on the web app, when only numerical ID is available. After receiving this feedback, first names and second names were added as entries, along with IDs, in order to make the data easier to navigate.

Participants were also asked the level of difficulty experienced when performing the tasks set. This was recorded as quantitative data and can be seen in the figure below.

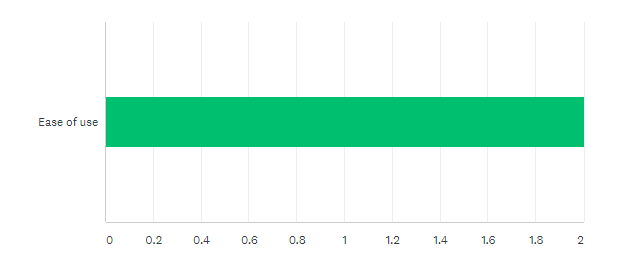


Figure 1: the ease of use experienced

Participants were also asked how in control they felt while performing the tasks, which was also recorded as quantitative data, and can be seen in the figure below.

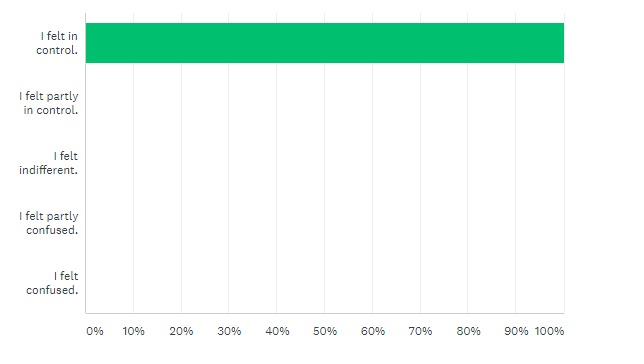
****

Figure 2: How in control users felt

#### Mobile app

In terms of the mobile cognitive walkthrough, the positive feedback received included that the use of the app was very straightforward, with the booking system pointed out as one of the easiest features to use. Users also reported an appreciation of the minimalistic and simple design, as well as some features of navigation.

However, one of the criticisms received was that a better navigation control system should be in place, and more meaningful identifiers should be present on buttons. With this feedback in mind, we implemented a sliding tab menu layout in the actual application, with a relevant skeueomorph for the opening of the tab, as well as relevant skeueomorphs for bookings, timetables, and account information.

When asked what features were, if any, the most difficult to use, the navigation to previous bookings was bought up. When asked for elaboration, it was found that this was due to confusing button names. This was later rectified to some extent, with more common skeuomorphs and icons, as well as more descriptive buttons. An example of this was the use of view bookings and bookings buttons being ambiguous, which was overhauled in the final product, allowing a swipe navigation of active bookings and completed bookings.

When asked if there were any features the users felt were missing application wise, a common response was a cancel ticket functionality. This was ultimately not incorporated into the final product, as time constraints limited the project in terms of developing a minimum viable product. Had time management been enforced more strictly, this would have likely been added to the project backlog and implemented as a feature of the system. Another feature that never got implemented, despite being suggested was a notification system for mobile app users to notify of replacement services. This was also not implemented due to time management issues.

Other factors that were considered were the ease of completing the tasks set, assessed quantitatively, on a scale of 1 to 5, with 5 being least easy, and 1 being most easy, with 100% of participants reporting extreme ease of use, as seen in the figure below.

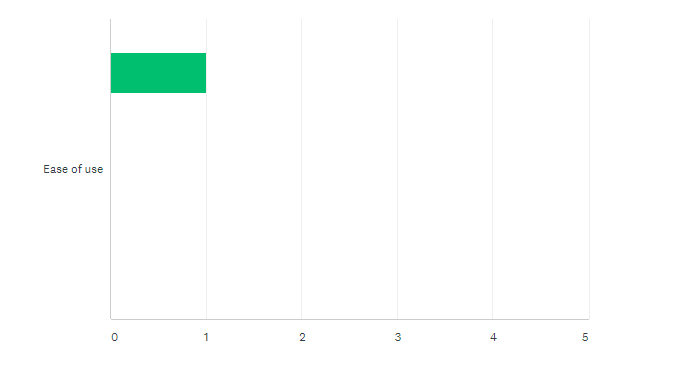


Figure 3: Ease of use.

Along with this, the overall experience rating of the use of the app was taken, with 1 being best experience, and 5 being worst experience, with a further 100% of participants reporting extremely good experiences, as seen in the figure below.

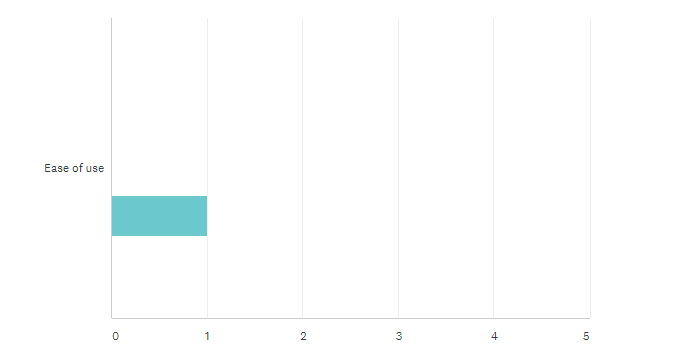
****

Figure 4: overall experience rating.

The final piece of data assessed quantitatively was how in control a user felt while using the navigation, with 100% of users stating they felt they were partly in control, as seen in the figure below.

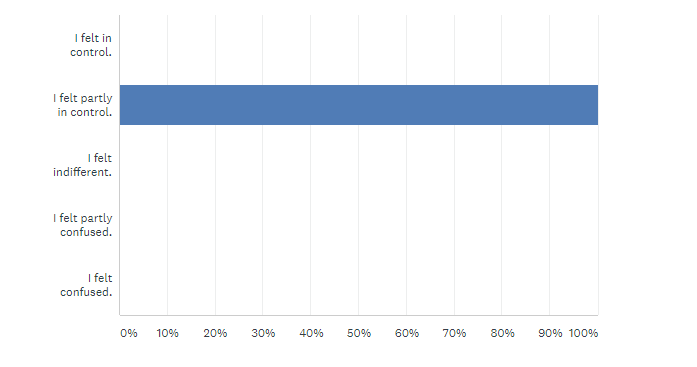
****

Figure 5: How in control users felt while completing tasks.

### Usability Test of the Mobile App

The usability test for the mobile app asked the same questions as in the cognitive walkthrough, to determine whether the now implemented changes suggested in the cognitive walkthrough yielded better results.

In general, positive feedback included that the app was straight forward to use, and intuitive. Ease of completing the tasks was typically valued at 2, with 1 being most easy and 5 being least easy, with overall experience with the application gaining a typical rating of 2 between 1 being mostly good and 5 being mostly bad also. Users also reported they felt in control of the application while conducting the tasks set.

One of the criticisms was the date picking function, with it feeling unintuitive and not having firm boundaries on what can and can’t be picked. This was then taken on board and changed so a time picker was implemented in order to make the time selection more restricted and intuitive. Another point of critique was not having a sense of a company name. This was rectified by adding the company name NationalCoach as the header for the app.

# Security

## Features Implemented

This section will cover security features we have implemented in the system and how the protect the user’s data both in transit and at rest and the retrieval of the data.

### SQL Injection

In terms of SQL injection, we have some protection against certain attempts of injection to drop tables and remove data. This is done by our use of the entity framework to access our database through the ASP.NET Web API. This allows the front-end systems to only access an image of the database and not the database directly so only REST operations are able to be performed. This means that the table structure cannot be changed by the API and since there is no direct access to the database the structure of the database cannot be changed by SQL injection attacks.

### Hashing and Salting

We have implemented hashing and salting algorithms on our middleware to secure our passwords when stored at rest on the database and when transited to the API for use. This makes the system less vulnerable to passwords being leaked if the database is accessed or data is intercepted in transit. The specific hashing algorithm implemented is SHA-256 along with a salt the length of 32 (this is a good size).

Also, passwords are never sent to the front-end applications meaning there is no way for them to be seen on the front end on accident. This counteracts sensitive data exposure which is one of the OWASP Top 10 vulnerabilities.

However, in our system we do have some limitations to what we have done with hashing and salting explained below.

### Token based Authentication

We have also added a token-based authentication system to our log in systems on the API so that when a user is authenticated, they are given a token to access further areas of the system such as data retrieval and manipulation. This means that any pages that would otherwise hold data that can be viewed by the end user will show nothing as they don’t have a token from authenticating through the official channels.

### Role-based authentication

We have added separate authentication for each of our client systems to distinguish between employees of the company and customers booking tickets. We have done this so that only customers can access the mobile application to make bookings and the employee systems, which contain sensitive data, can only be accessed by employees.

This allows for better data segregation and prevents incorrect or secure data being viewed by the wrong people and potentially being used maliciously.

## Features Not Implemented

The following are features that weren’t implemented into the system but if we had time or were to repeat the project, we would implement these features to enhance the security of the system.

### HTTPS

If we had more time to work on security for our system, we would like to enable HTTPS so that data is transferred across networks more securely. This would reduce the risk of data being intercepted and used as the data will be encrypted automatically during transit.

HTTPS, however, has been tested using IISExpress although due to the limitations of the University server we could not make use of HTTPS.

### Data in Transit – API

As mentioned above there are some vulnerabilities with our hashing and salting algorithms.

The main issue we have is that all of the hashing and salting is performed on the API as opposed to the front-end systems. This means that new account and log in passwords are being passed over a network to the API in plain text. This brings about the issue of passwords being intercepted in transit to the API with no decryption needed to expose them for malicious intent.

In future we would implement the hashing and salting algorithms on the client-side systems so that there is less risk of sensitive data exposure on our part.

# Software Engineering

## Use of Software Patterns

### MVC Pattern

One of the main software patters we have used throughout the entire system is the Model View Controller (MVC) pattern. We used this in both the web application as part of ASP.net MVC structure, and in the desktop application by having controllers handling the data manipulation and sending data to the GUI to be shown.

This allowed for good separation of logic and viewing code for high cohesion. This means that specific classes have their own specific purpose to perform so there is no redundant functionality that doesn’t belong to that class. This means that there is low coupling between sections of code so code can be reused more effectively in other areas or future projects.

### Singleton Pattern:

The singleton pattern is used mainly within the mobile and desktop application. It ensures that we can only get one instance of a specific class/object throughout it’s use in the program. Although it is a small pattern its use is essential within the mobile application which uses many asynchronous components.

## Good Areas

### API Logic

One of our strongest practices is that we kept most of our logic on the API including authentication, data filtering and security logic. This allows for good security as data is withheld when it is not required by the client such as passwords and other sensitive information.

This also means that there is low coupling between the client-side applications and the system logic meaning that the client systems primary function is the user interface and not much data processing in tern creating a very high cohesion system. Another positive of this is that this API can be consumed much easier by other systems without as many pre-requisites to be used making it multi-functional.

### Object Mapping

We also made good use of 3rd party object mapping software to handle the assignment of JSON data into class objects to be used by the client-side systems.

For the desktop application we used “Jackson” for the object mapping as it would do the HTTP request and mapping onto a java object all in one line of code when provided the correct endpoint to expose on the API. This allowed for simple conversion of JSON data sent from the API to objects that can be easily transferred into front end views for the end user.

In the website we used the .net built in “Newtonsoft.Json” libraries to manage all of the data sending and receiving by it using HTTP packets to receive all of the HTTP data allowing us to verify successful requests by looking at the status code allowing for better error handling in the case of problems occurring with requests.

When it comes to the mobile application, we used Volley (Google) for our object mapping as it was simple to handle JSON data conversion onto java model classes housed in the mobile application. The good thing about using Volley is that the request is done asynchronously by default so data that is retrieved and seen by the end-user is always accurate to what is on the database thus providing a much better user experience for the customer who is then more likely to use the app more.

### Expandability

Our system is very expandable in the respect that there is a high amount of inheritance of functionality from parent classes. This is extremely prevalent with the API connection classes only having generic methods for REST request to endpoints on the API. This meant that individual classes had to inherit those methods to be able to perform requests to the API.

This also means that if further endpoints are added to the database and the API, they can easily be accounted for in client-side applications by adding a new model class and controller to handle the data using the existing connection framework.

## Shortcomings of the System

### Multi-user authentication

We would have liked to have implemented a separate section of the website to be for the manager as they would have different privileges and responsibilities than an administrator. However, the issue we encountered was implementing a distinction between users when authenticating on the API and then directing the end user to the right section of the website.

Also, our navigation bars layout was universal across all pages of the website meaning that we couldn’t get access privileges to be restrictive enough to be effective, so we moved the user stories from the manager to be associated with the administrator’s.

### Additional Functionality

We also have much less functionality than we had anticipated at the start of the project. This is partially down to the rate of which we were developing software being slower than we had initially anticipated due to our inexperience with some of the architecture we had to use.

Also, we had a large amount of user stories established in our requirements phase of the project. This in hindsight should have been reducing to make a more simplistic system that had a more focussed set of functionalities to a higher standard. Not changing this focus until the end of the project caused a large amount of problems when implementing the more complicated user stories at the end, causing them to be rushed and of lower standard.

### Code Reviews and Testing

One of the major areas of software development that we didn’t complete would be any kind of review and testing of our code. Although code reviews were performed informally as frequently as possible, we have not made formalised the process.

This was down to time constraints and a lack of understanding of how unit tests work in both Android Studio and ASP.NET.

This means that there could be bugs that unit tests could pick up and that could be resolved. This would create a more fool proof service that isn’t prone to unusual bugs that could be in the underlying system.

Also, as we haven’t carried out formalised code reviews, there could be poor practices being employed by us that wouldn’t get picked up. This could cause our system to have unknown optimisation or security issues that could be exploited by the public.