PRCS252E - Final Report

Contents

[2 Project Management 4](#_Toc7784121)

[2.1 Team Project Management 4](#_Toc7784122)

[2.2 Changes during the project 4](#_Toc7784123)

[2.3 Risk Analysis 4](#_Toc7784124)

[3 Requirements Analysis and Design 5](#_Toc7784125)

[3.1 Product Backlog 5](#_Toc7784126)

[3.2 Use Case UML Diagrams 8](#_Toc7784127)

[3.2.1 Administrator 8](#_Toc7784128)

[3.2.2 Driver 9](#_Toc7784129)

[3.2.3 Customer 11](#_Toc7784130)

[3.2.4 Manager 12](#_Toc7784131)

[3.3 Changes during Development 12](#_Toc7784132)

[4 Database Design 13](#_Toc7784133)

[4.1 Database Diagram 13](#_Toc7784134)

[4.2 SQL Create Statements 14](#_Toc7784135)

[4.2.1 Employee Table 14](#_Toc7784136)

[4.2.2 Employees Archive 16](#_Toc7784137)

[4.2.3 Shifts 17](#_Toc7784138)

[4.2.4 Customers 18](#_Toc7784139)

[4.2.5 Stops 20](#_Toc7784140)

[4.2.6 Routes 21](#_Toc7784141)

[4.2.7 Routes Archive 22](#_Toc7784142)

[4.2.8 Route Stops 23](#_Toc7784143)

[4.2.9 Coaches 23](#_Toc7784144)

[4.2.10 Coaches Archive 25](#_Toc7784145)

[4.2.11 Journeys 26](#_Toc7784146)

[4.2.12 Bookings 28](#_Toc7784147)

[4.2.13 Booking History 30](#_Toc7784148)

[4.2.14 Replacements 31](#_Toc7784149)

[4.3 Normalised Tables 33](#_Toc7784150)

[4.3.1 First Normalised Tables 33](#_Toc7784151)

[4.3.2 Second Normalised Tables 34](#_Toc7784152)

[4.3.3 Class Diagram Comparison to Normalised Tables 35](#_Toc7784153)

[5 Usability 36](#_Toc7784154)

[6 Security 37](#_Toc7784155)

[6.1 Features Implemented 37](#_Toc7784156)

[6.1.1 SQL Injection 37](#_Toc7784157)

[6.1.2 Hashing and Salting 37](#_Toc7784158)

[6.1.3 Token based Authentication 37](#_Toc7784159)

[6.2 Features Not Implemented 37](#_Toc7784160)

[6.2.1 HTTPS 37](#_Toc7784161)

[6.2.2 Data not encrypted until the API 38](#_Toc7784162)

[7 Software Engineering 39](#_Toc7784163)

[7.1 Use of Software Patterns 39](#_Toc7784164)

[7.2 Good Areas 39](#_Toc7784165)

[7.3 Shortcoming of the System 39](#_Toc7784166)

[7.4 Assumptions 39](#_Toc7784167)

# Project Management

## Team Project Management

* Sprint user stories were allocated so that each team member got equal attention to all applications

## Changes during the project

* We had to prioritise and reduce the amount of user stories to complete during the middle stages of the project due to the development process taking longer than expected

## Risk Analysis

* In future be more prepared and experienced with the software to be used on the project to allow for a more efficient development process without stopping to learn new things.

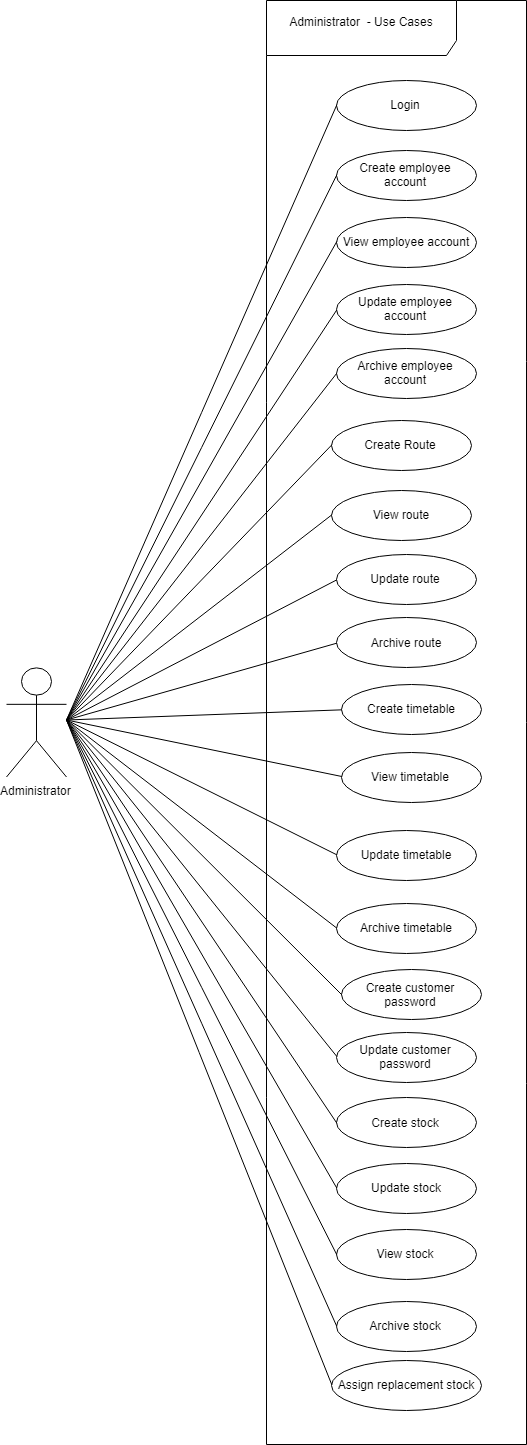
# Requirements Analysis and Design

## Product Backlog

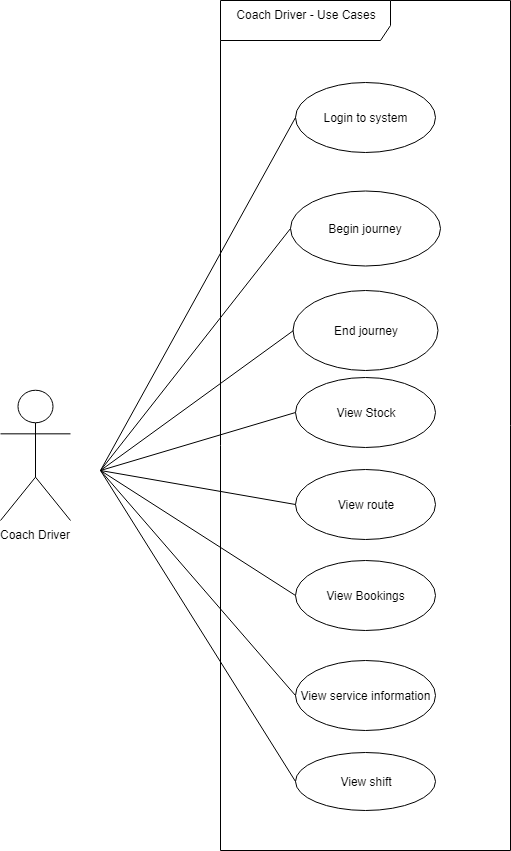
|  |  |  |
| --- | --- | --- |
| Backlog | In Progress | Complete |
| [DRIVER - DESKTOP] As a driver, I want to be able to accept an e-ticket booking reference from a passenger, to validate their journey on the coach. | [MANAGER - WEB] As a manager, I want to be able to assign a driver a coach for the shift, so that the driver can perform his duties. | [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. | [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 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 create routes, so that coaches have a set path. | [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 | [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 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. | [DRIVER - DESKTOP] As a driver, I want to be able to view coach capacity, so that the maximum number of passengers can be ascertained. | [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 update journeys, so that I can view the routes and times. | [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. | [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 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. | [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 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. | [DRIVER - DESKTOP] As a driver, I want to be able to access the shift schedule, so that I know when I need to work. | [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. | [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 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. | [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 create journeys, so that the routes and times can be documented. |
|  | [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 be able to archive coaches in the stock inventory, so they can be decommissioned if not in use or replaced. |
|  | [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 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 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 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 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 be able to update employee accounts, so that information related to employees can be changed. |
|  | [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 create employee accounts, so that new employees can undertake their duties. |
|  |  | [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. |

## Use Case UML Diagrams

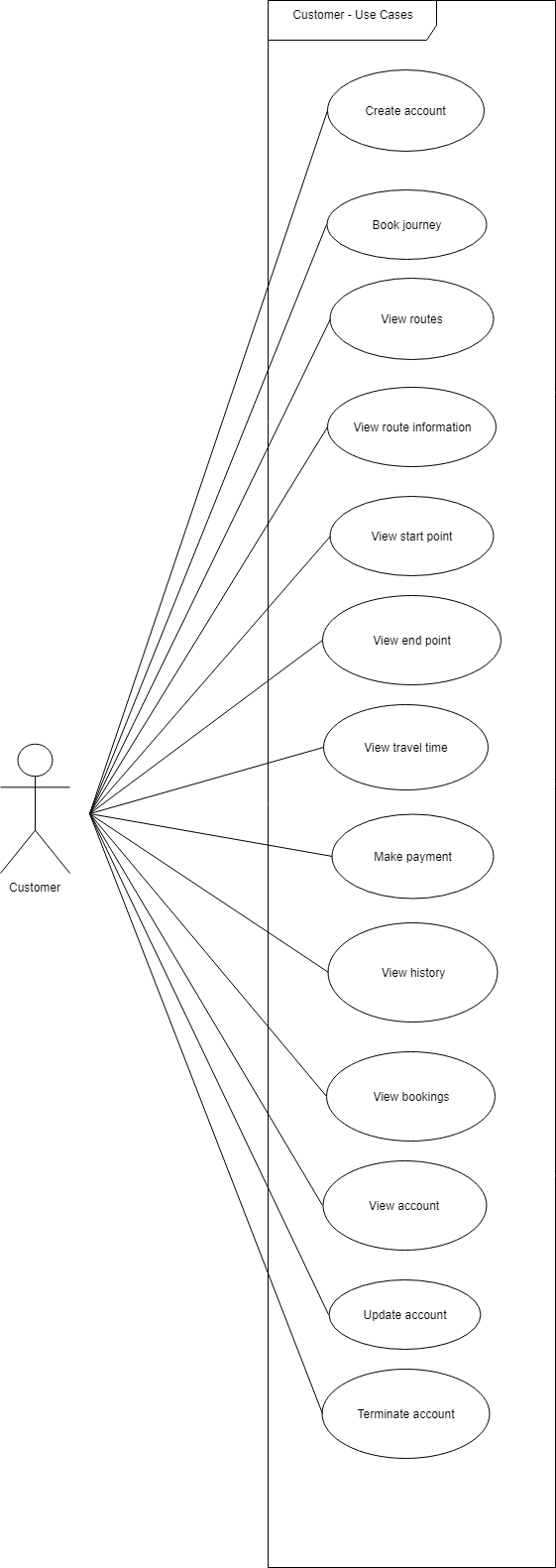
### Administrator



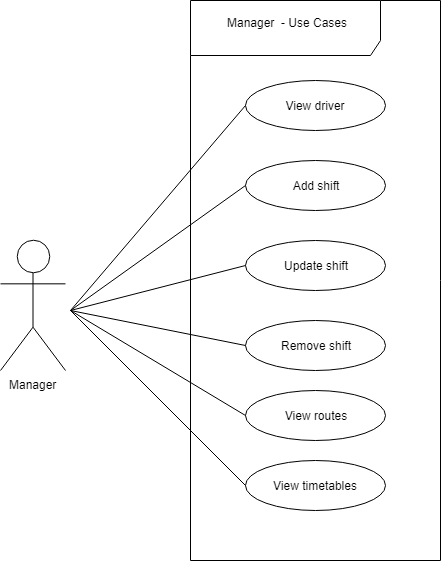
### Driver



### Customer



### Manager

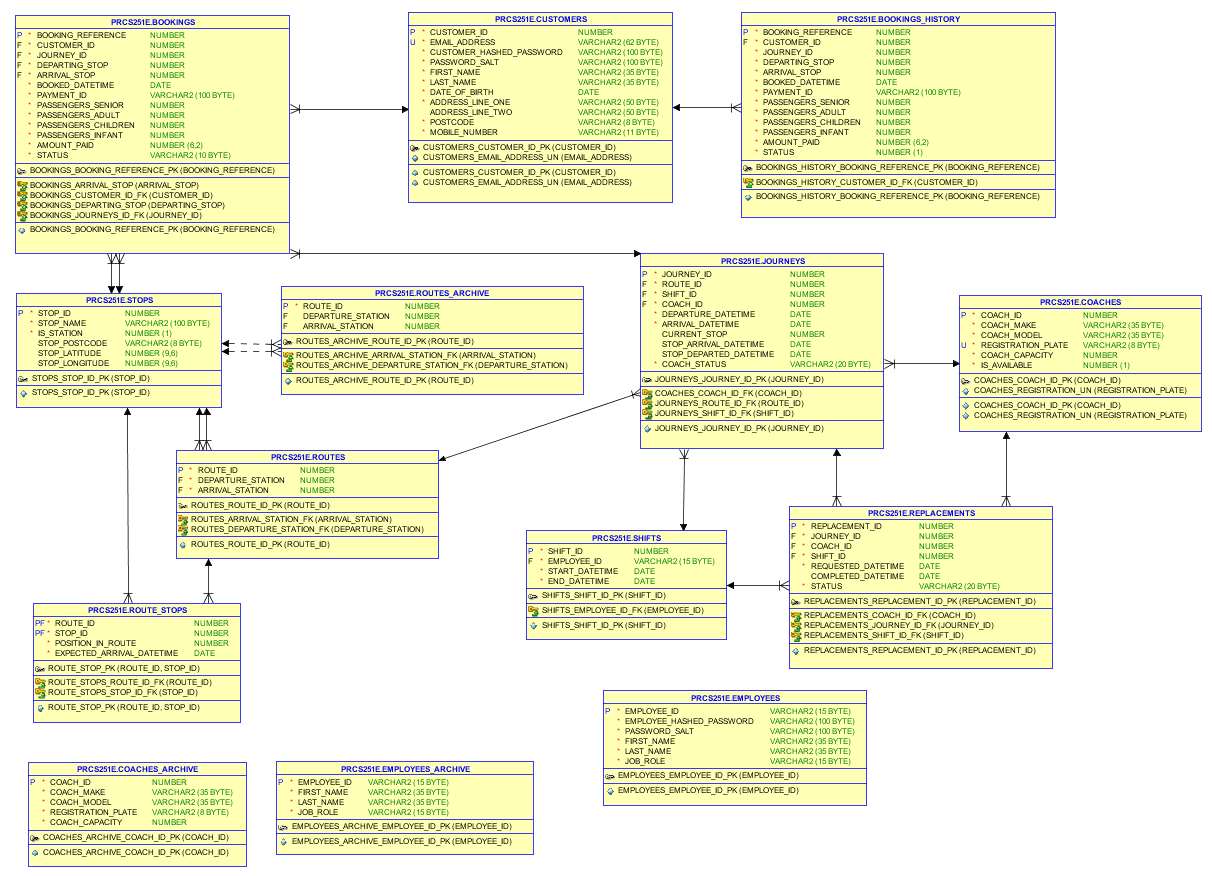


## Changes during Development

* Removal of Manager user – user stories converted to admin
* User story removed – 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. This was removed as the driver doesn’t need to know how many coaches are available, they just need to know what coach they will need.
* User story 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. Changed to adding a new shift for a driver.
* During sprint 4 the core functionality was decided, and these were user stories that have highest priority to be completed
* Timetables table was removed from the database and journeys included the timetable elements

# 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

* A large amount has been added to the database compared to what has been normalised
* Tables have been added since the last normalisation attempt and some have had name changes or extra attributes included
  + Added tables
    - Bookings
    - Bookings history
    - Routes archive
    - Route stops
    - Coaches archive
    - Replacements
  + Table name changes
    - Stations table has been changed to stops

# Usability

# Security

## Features Implemented

This section will cover security features we have implemented in the system and how the protect the users data both in transit and at rest and also 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.

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.

## 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.

### Data not encrypted until the 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 front end systems so that there is less risk of sensitive data exposure on our part.

# Software Engineering

## Use of Software Patterns

* MVC pattern heavily

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.

* Observer?

## Good Areas

* Kept most of the logic on the API
* Good use of object mapping 3rd party software’s to handle API connection and map data onto classes to be used by the GUI – comments on object mapping in mobile app

## Shortcoming of the System

* Multiple users on web app – manager
* Liked more user stories to be completed to have a more complete system
* Code reviews

## Assumptions