PRCS252E - Final Report

Contents

[2 Project Management 4](#_Toc7526128)

[2.1 Team Project Management 4](#_Toc7526129)

[2.2 Changes during the project 4](#_Toc7526130)

[2.3 Risk Analysis 4](#_Toc7526131)

[3 Requirements Analysis and Design 5](#_Toc7526132)

[3.1 Product Backlog 5](#_Toc7526133)

[3.2 Use Case UML Diagrams 5](#_Toc7526134)

[3.3 Changes during Development 5](#_Toc7526135)

[4 Database Design 7](#_Toc7526136)

[4.1 Database Diagram 7](#_Toc7526137)

[4.2 SQL Create Statements 8](#_Toc7526138)

[4.2.1 Employee Table 8](#_Toc7526139)

[4.2.2 Employees Archive 9](#_Toc7526140)

[4.2.3 Shifts 11](#_Toc7526141)

[4.2.4 Customers 12](#_Toc7526142)

[4.2.5 Stops 14](#_Toc7526143)

[4.2.6 Routes 15](#_Toc7526144)

[4.2.7 Routes Archive 16](#_Toc7526145)

[4.2.8 Route Stops 16](#_Toc7526146)

[4.2.9 Coaches 16](#_Toc7526147)

[4.2.10 Coaches Archive 18](#_Toc7526148)

[4.2.11 Journeys 19](#_Toc7526149)

[4.2.12 Bookings 21](#_Toc7526150)

[4.2.13 Booking History 23](#_Toc7526151)

[4.2.14 Replacements 24](#_Toc7526152)

[4.3 Normalised Tables 25](#_Toc7526153)

[5 Usability 27](#_Toc7526154)

[6 Security 28](#_Toc7526155)

[6.1 Features Implemented 28](#_Toc7526156)

[6.2 Features Not Implemented 28](#_Toc7526157)

[7 Software Engineering 29](#_Toc7526158)

[7.1 Use of Software Patterns 29](#_Toc7526159)

[7.2 Good Areas 29](#_Toc7526160)

[7.3 Shortcoming of the System 29](#_Toc7526161)

[7.4 Assumptions 29](#_Toc7526162)

# Project Management

## Team Project Management

## Changes during the project

## Risk Analysis

# Requirements Analysis and Design

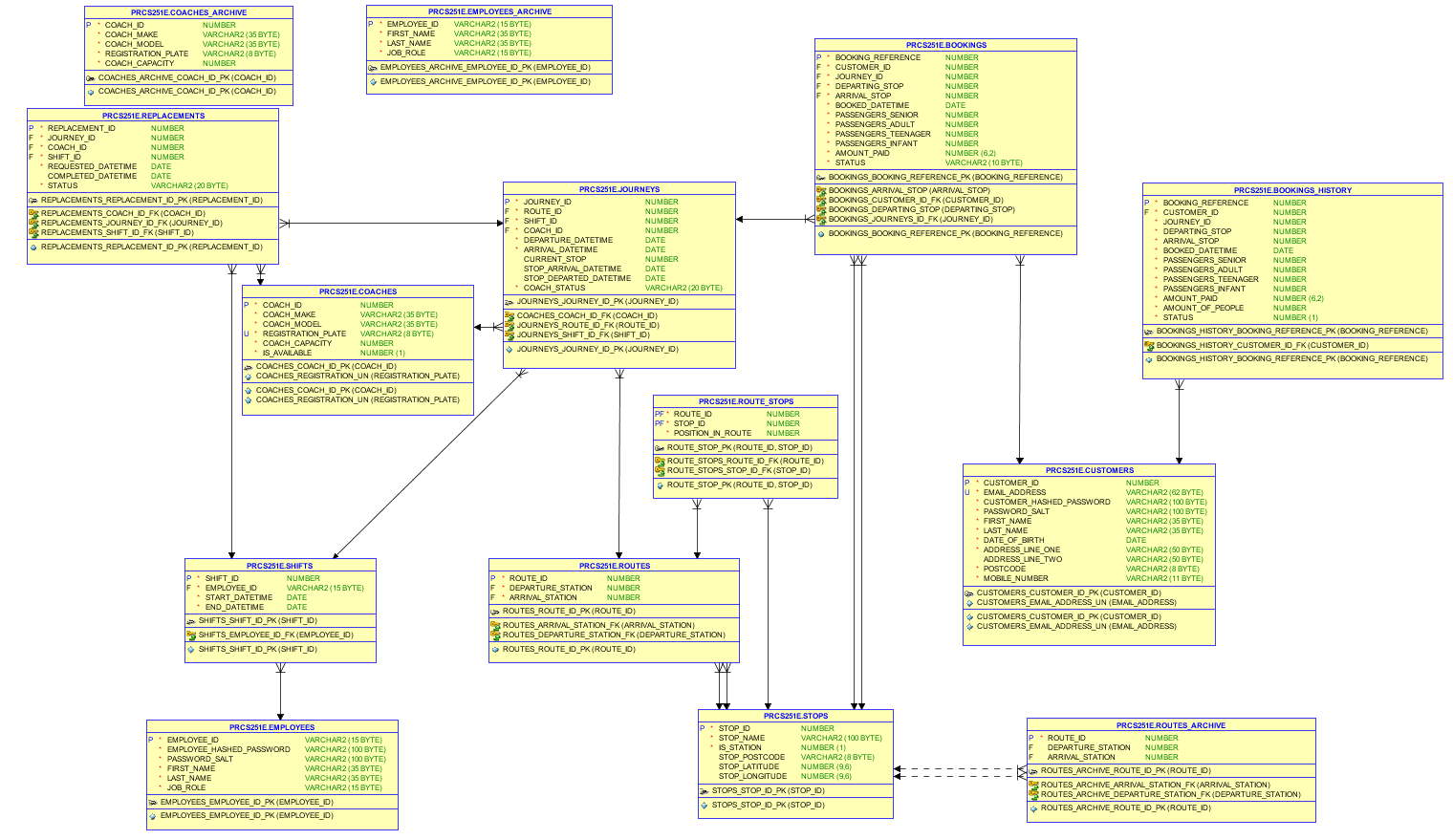
## Product Backlog

## Use Case UML Diagrams

## Changes during Development

# 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

# Usability

# Security

## Features Implemented

## Features Not Implemented

# Software Engineering

## Use of Software Patterns

## Good Areas

## Shortcoming of the System

## Assumptions