**Mini Project Script**

**Main Screen**

Enter 1 – 4 or raise an error

**Product Menu**

Enter a valid input or errors out

**Print screen**

Yes or No – error if enter incorrect input

Updates in real time from mysql database

**Create a product**

Press 0 if in by accident

Hawaiian

Price – input error validation – enter string

**Update price**

ENTER A VALID ID ONLY!!

Show updated price

**Replace Product**

ENTER A VALID ID ONLY!!

Show updated details

**Delete a product**

Enter wrong ID – shows error handling

Delete and then show

**Courier Screen**

Enter a valid input or errors out

**Print out list**

Updates in real time from database

**Create Courier**

Enter Name

Phone number validation

Print out new courier

**Update Courier**

ENTER VALID ID ONLY

Number validation again

Updated in menu

**Replace courier**

ENTER VALID ID ONLY

Name and number

Print update

**Delete Courier**

Enter wrong ID – shows error handling

Delete and then show

**Order Screen**

**Print out orders**

Explain list and that order specifics are in next section

**Create Order**

Name

Address

Number validation

List of couriers – inc validation

Products – validation

Mention status automatically set to received

Show in print menu

**Update Status**

ENTER VALID ID ONLY

Enter a status change

Print out change

**Replace Order**

ENTER VALID ID ONLY – CHOOSE OPTION THAT IS RECEIVED

Explain if you do not want to change a value, enter through it

Name – can tab

Address – can tab

Number – can tab

Courier – can tab

DO NOT TAB THROUGH ORDER STATUS – CHANGE TO ORDER PREPARING

DO NOT TAB THROUGH PRODUCTS

Print out list

**Delete Order**

Can enter wrong ID

Delete and show

**Reports Screen**

**Details for a specific order**

Order validation

Choose 13 / 26 and another

**Orders not yet delivered**

Any orders that are either received or preparing

**Delivered orders**

**Cancelled orders**

Explain based on their order status

**Complete Order Status**

Explain chart

**Deliveries per courier**

Explain chart

**Back End**

Split app into several modules…

**App module**

Contain all menus for app

Generic menu for products, orders and couriers

CRUD menus

Functions are usually split into generic product/courier and a separate order one

Return functions

Run app from here

**Util**

Several utility type functions

**Database**

Similar to others, created a connect to database function and generic sql select and execute functions

**Printing**

Function that pulls the information from my database and returns it in list form

Depending on which database is used, different headers will be assigned to create a table using the tabulate utility

Generic table outline for my reporting section

**CRUD**

Generic add for products and couriers

Add order is separate, includes a couple of functions to assign couriers and products

Replace products and couriers is housed within same function

Replace order is larger and includes ability to pass through option if required

Update is generic for all. The existing IDs line ensures that only a viable id can be entered.

Additional functions include option to change the order status, assign a courier from a list and assign products to an order.

**Reporting**

R1 – uses joins to return name, address and products for a given order in tuple form.

Use a for loop to build a list of products

R2, R3 and R4 use a select query regarding the order status and will use for loops to fill lists of the name, address and order status depending on the criteria entered

R5 grabs a list of the order status and counts each order type. This information is placed into a basic pie chart.

R6 grabs a list of courier names and the list of couriers assigned from the orders database.

The number of times a courier has been assigned is counted and the names and number of times a courier has delivered are plotted on to a basic bar chart.

**Testing**

Testing has been challenging and a part that I have shyed away from – this is obvious in the design of my code.

I have a great respect for Test Driven Development of code as Non-TDD has caused several problems which has led to not being able to complete the full range of tests I wanted to.

I have a number of tests that do not work as they have an oddity that runs functions in the my app – this led to a surprisingly high number of hat’s added to my product database and couriers named Bob to my courier list – as my mocks and patches were not operating correctly.

It has become abundantly clear that to try and make every function self-contained rather than interdependent on other functions is a corner stone of easy unit testing – sadly, my appreciation of this came late in the day which did not allow me to re-factor my code.

Add new product to database code worked thanks to a number of people and I have several tests that ran as the function was self-contained. This is one of my biggest learnings from the past several weeks – that and the fact that it does not help to shy away from areas you do not fully grasp. If I had taken the time to understand testing, I would have understood the importance of TDD which would have made testing a much easier operation.

**Personal experience**

Rollercoaster of emotions ranging from doom and gloom up to, I know exactly what I am doing – does not last long.

I look forwards to working out a problem and get an. Amazing feeling of success when overcoming a challenge both big and small.

I am most proud of hitting a wall and having to refactor – deleting code that has taken days and weeks to write as I have found an improved way (usually significantly shorter) to carry out the same functionality. Soul destroying followed by unadultered joy.

On a personal note, the group, both instructors and boot campers have been incredibly supportive, everyone helping each other out – the overall comradery and vibe from the group is incredibly positive and it has been a real please to work alongside the team.