WJEC A Level Computer Science Unit 5 Project

Jack Gillespie

Discussion

About the Organisation

Carrick Technical Solutions is a small gardening firm, with three employees. The company consists of Shaun Gillespie, the director, as well as two assistant employees. CTS specialises mainly in lawns and hedges, however it offers a wide variety of services, including indoor tasks such as painting and construction. Whilst the company and majority of its clients are in Wrexham, it also has clients in Flintshire and Cheshire.

Problem

The majority of quote calculations are done on paper and/or mentally (depending on the complexity of the quote), taking into account factors such as: time, type of job, any materials required, etc. This makes the system liable to human errors, such as miscalculations and/or using incorrect data, and so prices can often be quite larger or smaller than the expected result. This can be an issue for the company, as overquoting can result in a loss of service, and underquoting will result in the company losing money, as its earnings will be lesser than the required amount to cover the cost of its expenditures.

Booking is also done manually, into a book, creating a timetable of customers. The customers are organised on the timetable based off of location to minimise transport time. The order in which customers are sorted is organised manually by the director, in an attempt to minimise the travel time between customers, by grouping customers from the same areas together. Days are often missed due to poor weather, resulting in customers having to be fitted into vacant slots on other days, requiring a recalculation of the timetable every time it rains (which is fairly often in the UK). As the bookings are done manually, it can be a time consuming process to order the customers into a timetable, especially since re-ordering customers missed due to poor weather is a common process. And, as this task is done by a human the timetable may not be the most efficient version possible.

As well as these issues, there is also the minor issue that comparing data, such as one month's earnings with another's can be a slow process, as each booking from said month has to be manually found in the storage book, and added manually. This is also prone to miscalculations/misreading data errors.

Another minor issue is storage. As all of the company's information is stored on paper, it takes us physical space to store, and as some of the information has to be kept for a matter of years, a large amount of physical space has to be dedicated to

storage. And as well as this, the information has to be kept secure under lock and key, meaning fairly large secure cabinets have to be utilised.

Objectives

The primary objective of this project is to resolve all of the aforementioned issues that exist within the current system, whilst also ensuring the solution has a simple and easy to use design, to prevent overcomplication. The system will be secured through the use of a password-protected login system. This login system will be encrypted, to ensure security, as well as utilising different layers of user access, to prevent unauthorised employees from accessing confidential information.

In order to resolve the issue of inaccuracy in quote calculations, I will implement an inbuilt calculator into the program. As the computer will perform the calculations this will eliminate the possibility of arithmetic errors. The program will also automatically read as much information as possible, to minimise the amount of user-inputted information to prevent accidental mistakes in data input. All input fields will utilise several layers of validation (presence, length, range, data type and format checks) and verification to also try to ensure this. This should minimise the opportunity of human error when using the system.

In order to resolve the issue of inaccuracy and inefficiency when calculating timetables, my program will automate this process. As this process will be automated, the possibility of human error is removed. This should also be faster for the computer to calculate than a human, saving the director time. And, as the ‘route’ is being generated by a computer, it can calculate the efficiency of the route to ensure that the most efficient route has been chosen.

In order for the program to have the required information to perform this task, the program will need to allow for users to input customer details (vitally, the customer’s address) and booking details (vitally, the job length).

This routine of calculating the timetable can be rerun to recalculate the timetable to fit in missed customers due to weather preventing work.

Data comparison can be automated, as the program can search for data faster and easier than a human, as well as being able to calculate the value of all data found without arithmetic error. The computer can also generate graphs and other visible representations of data output, to help the user better understand the data.

Using a computerised system solves the storage problem, as no physical storage is required other than the computer system itself. And data can easily and securely be backed up onto external hard drives.

User Requirements

The system will utilise a user-access-level system, to ensure ‘high end’ functions can only be performed by administrative employees.

High level users, such as the director will need to be able to add bookings to the system, assign employees to bookings, view predicted earnings and compare earnings, calculate quote prices, generate a new timetable, view the timetable, and add new users to the system.

Low level users, such as the assistants, will only be able to calculate quote prices and view the timetable.

Limitations

There will be some limitations to this system. This system will be offline only, not using servers to allow access from external devices. Workers must be applied to jobs manually, as it will be too complicated to do automatically. This program will only run on the Windows OS, and not alternatives such as Linux and OSX. The system will need to comply with the Data Protection Act 1998 and General Data Protection Regulation. And, even though the opportunity is minimised through the aforementioned systems, human errors will still be possible, which can compromise the accuracy and integrity of the system’s calculations and functions.

Feedback

I discussed my proposed system using a presentation which was presented to a group of pupils within my class, who evaluated every aspect of the proposed computer system and this resulted in valuable feedback that will allow me to refine my initial ideas.

The first piece of feedback was in regards to the storage of files, and whether data could be backed up. As the data will be stored as .txt files, they can easily be copied onto external hardware to be backed up, and then re-added to the system folder if the initial data is lost, and will work seamlessly with the system (provided the file is not edited). However, this system is not the simplest, and could be problematic as the direct handling of files can result in the files being deleted and lost if performed wrong, and so integrating a backup system into the program could be beneficial.

The second piece of feedback I received was about one of the limitations I mentioned: that the system will be offline only. Whilst it could be possible to develop the system to work online and be remotely accessible, it would be a lot of complicated work to integrate. As well as this, it would open up the system to remote access, creating the potential issue of hacking and unauthorized access, meaning the security system would have to be greatly overhauled and improved. All of this would be harder and slower to create the program, as well as require more

maintenance to run, which does not suit my client's needs. As well as this, an online system would effectively be a waste of resources, as all of the company's employees work out of one office with a single computer system, and so the only time they might need to access the system remotely would be to view the timetable when working. However, viewing the timetable in the field can be easily solved using paper printouts.

The question of 'will more than one user be able to use it at a time, and if not, could a dedicated PC be set up as a central server?' was also asked. The system will only be usable by one user at a time as there will only be one copy of the system on a single device. This device could be setup as a central server, accessible from other devices on a LAN network to allow multiple devices to access the system -- however this will also be an unnecessary complicated that requires maintenance, as there is only one PC in the office, and so there are no external devices to connect to the central device even if it was setup to be capable of handling it.

I was also given feedback as to the nature of usage of the validation protocols used in the system. Presence checks will be utilized on all essential fields to prevent the lack of essential data causing issues. Length checks will be stored on every field that needs to be externally stored, as the file system will be using fixed length fields, and exceeding this could be problematic. Range and data type checks will be used for numerical values, to ensure that the numbers are able to be used in calculations as they will actually be stored as numbers, as well as the numbers being an acceptable value. Format checks will be used on certain fields to ensure that the correct data has been inputted (such as postcodes having to be in the form AA000AAA, or date having to be DD/MM/YYYY). Verification such as double-inputs will also be used for the creation of new users' passwords.

One of my peers made the good suggestion of implementing a pop-up reminder system into the program, to remind users when their bookings are. As the users will not have the device with them as they are out working, they will only be using the system early in the morning, or later in the evening, as a real-time reminder system would not be overly useful to them. However it could be useful for the director, as the evening before it could remind him what type of jobs (mowing, hedges, painting, etc) are booked for the next day, so that he knows what to pack in the van (lawn mower, hedgecutter, paint brushes, etc.)

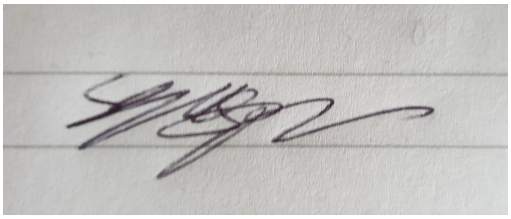
The last piece of feedback I received was asking how the system would display predicted earnings to the user. When displaying a single value (for example the amount of money earned on the week of 01/01/2020) the system will just display it as a numerical value (£500), whereas when displaying multiple values (or example the amount of money earned on the week of 01/01/2020, 08/01/2020, 15/01/2020 etc) it will use a graph to visually display the differences in the values.

Investigation

Interview

For the interview, I spoke to Shaun Gillespie, the director of Carrick Technical Solutions Ltd.

Jack: Hello, Shaun. Thank you for agreeing to participate in this interview about the company and your current system.
Shaun: Hi Jack. No problem. I am happy to help where I can by answering your questions.
Jack: Thank you. So what is the purpose of your current system, and how does it operate?
Shaun: Our current system is used to schedule and keep track of work on a weekly basis. A list of regular customers is kept, along with details such as their addresses, the job details, etcetera. A separate list of 'one-off' irregular jobs is also kept, along with the same details. These documents are used to organise and schedule bookings. We also create and use quotes for large jobs, or smaller jobs for new customers.
Jack: And all of these files are paper-records, yes?
Shaun: That is correct. All of these documents are handwritten on paper, and stored for varying amounts of time, depending on what is on the document.
Jack: Could you please elaborate on the timetable creation process?
Shaun: Of course. We have a rough standard fortnightly timetable. Once a week, on a Sunday, I go through the list for the next week, and will amend it where necessary, for example, adding in missed customers from last week, or any bookings from the 'one-off' list. The customers are organised by area to reduce drive-time between appointments.
Jack: Thank you for the details. Would you say there are any problems you are having with the current system? And if so, how are they impacting your company's productivity?
Shaun: At the moment, preparing the timetable for the next week can be a pretty time-consuming process, whereas if it was automated it could save countless hours over the course of the year. On top of this, searching for old records, such as month old quotes, can be quite time consuming, however I imagine that a computerised system would resolve this issue too.
*Signed and approved by **S. Gillespie***



Document Analysis

CLS.	63×38×2.4	31 OFF	@ £2.75 ea.	85.25
	63×38×1.82.4	11 OFF	@ £2.75 ea.	30.25
	70×45 C16.			
TIMBER	75×50×2.4	5 OFF	@ £5.69 ea.	28.45
	75×50×3.0	4 OFF	@ £7.85 ea.	31.40
	75×50×1.8	5 OFF 3.6 m	@ £9.41 ea.	47.05
CHIPBOARD	2.4×0.6.	13 OFF	@ £ 9 ea.	117
PLASTERBOARD	2.4×1.2×12.5mm	12 OFF	@ £7.60 ea.	91.20
	2.4×1.2×12.5mm	2 OFF	@ £7.60 ea.	15.20
	1.8×0.9×10mm	3 OFF	@ £7.85 ea.	23.55
INSULATION.	19.2 m² WALLS.	2.4×1.2×50mm 7 OFF	@ £39 ea.	273
BOARDS.	14 m² FLOOR.	5 OFF	@ £39. ea.	195
(41.9m²)	8.7 m² CEILING.	3 OFF	@ £39. ea.	117
				585
= 76 OFF 1.2×0.455m = £266 @ £3.50 each - £486 @ £6.40 WICKES				
INSULATION ROLLS	2 OFF 100mm.		£16.50 ea.	33
				1087.35
				−10% £978.61

DPM 4×12.5m. £32
PLASTER £6.50 MAYBE 6 OFF.
SCREWS, £30 ?
PLUMBING ?
ELECTRICS. £50 ?
PLASTER BOARDS. £10 ?
WC / CISTERN £80 ?
DOOR FRAMES £50 ?
DOORS ?
DOOR FURNITURE. ?

Prepared by	Shaun Gillespie - Director, Gardener
Purpose	To provide a price for the set job (including a breakdown of materials, man hours, etc.) primarily for the company, however it can also be used for the customer, should they want a breakdown of the price.
Size	A4
Number of parts	1
How often?	Created for each new/irregular job.
How many?	1
Medium	Paper
How long is it kept?	6 months

Data	***Type***	***Size/Example***	***Occurrence***	***Range***	***Source***
Name of Expenditure	String	Chipboard		N/A	Written by Shaun
Type of Expenditure	String	2.4x0.6		N/A	Written by Shaun
Price of Type of Expenditure	Real				
Quantity of Expenditure	Real	13		1 - 20	Written by Shaun
Price of Expenditure	Real	117		£1 - £500	Calculated by Shaun
Total Price	Real	978.61		£10 - 2000	Calculated by Shaun

M	T	W	T	F
8-9 SUTCLIFFE	7.45-9.15 BONAPARTE	NEIL. SECKA	MILLS	HT.
9-9.30 NIETROP	9.15-10.15 BURTON.	JILL	HOWGATE	
9.30-10 LEWIS.	10-15-11 HOLMAN	NEXT HARDEN		
10-10.30 CERYS	11-11.30 KATE.	HUGHES.		
10-30-11.15 ROBERTS	11.30-12 TAYLOR	SALES.		
11.15-12 MANDSLEY	12-12.30 BENNIONS.	KELLY		
12-12.30 HOWIE	12.30-1 GOODWICK.	THOMAS		
2-4 MELIA.	1.45-2.15 PRESICOS	DON.		
12.30-1 NICK.	2.15-2.30 READ	[illegible].		
1-1.30 BETTY (TREES)	2.30-2.45 REA	CANTWELL.		
1.30-2. KIRSTY.	2.45-3.15 CAPPER	LOW.		
4.30-5 WYNNE.	3.15-3.30 NEXT	DAWSON. ASHWORTH.		
4-4.30 WEDLEY.	3.30-4.30 HATTON			
DARK 5.15	4.30-5 BEV.			
	5-5.30 JONES			
	5.30-6.30. WILLIAMS.			

Document number	Not Numbered
Document name	Timetable (New)
Prepared by	Shaun Gillespie - Director, Gardener
Purpose	To provide employees with a clear schedule for the week.
Size	A4
Number of parts	1
How often?	Created once a week.
How many?	1
Medium	Paper
How long is it kept?	2 years

Data	***Type***	***Size/Example***	***Occurrence***	***Range***	***Source***
Customer Name	String	Ashworth	*Many	N/A	Written by Shaun
Day of Booking	String	Monday	5	N/A	Written by Shaun
Time of Booking	Real	8 - 9	*	8 - 18	Written by Shaun
Time of Sunset	Real	5.25	1	3.75 - 8	Written by Shaun

OCT

M 19	T 20	W 21	T 22	F 23
8-10 LMH 30	MOT	SECKA.	8-8.30 MORRIS 8-30 KEITH 9 (20)	HT.
10.15-11.15 GORE 50	9-9.30 MORRIS	JILL	9-9.30 FOSTER. 9.30-10.30 YORK	
11.15-12.15 JONES 70	9.30-10.15 KELSAL	NEXT HARDEN.	DYMORE 10.30-11	
12.45-2.45 DAVIS 110	10.15-10.45 FOSTER.		HEENAN 11-12.30 LUNCH 12.30-1	
2.45-3.15 MANDY 125	10.45-11.45 YORK.	(DYMORE)	ANN JONES 1-3 ASHWORTH 3-5 KEITH 5-6	
3.15-4.15 CAROLINE 155	12-12.30 KIRSTY WEEDS	HEENAN		HT
4.15-4.45 NEXT 170	1-1.15 READ	CANTWELL		READ
5-5.30 KIETH 190	1.15-1.30 REA	DAWSON.		REA
	1.30-1.45 NEXT	ASHWORTH.		NEXT
	1.45-2.15 CAPPER.			CAPPER
	2.30-3.30 HATTON			HATTON
	3-30-4 BEV			BEV
	4-4.45			JONES
	4.45-5.15 JONES			WILLIAMS
	5.30-6.30 WILLIAMS.			

THELMA HEDGE.	
	KIRSTY WEEDS
CAROLINE PAINT	BONAPARTE BANK PAINTING ETC MOLYNGTON
MELIA SHRUBS.	
KIRSTY TREES.	
KATE SHRUBS	
AUBREY HEDGE	
PHIL JONES.	
KIRSTY HEDGE.	
BENNIONS HEDGE	
RODDEN	
EVANS HEDGE.	
THOMAS TREES.	
DAWSON.	

Document number	Not Numbered
Document name	Timetable (Old)
Prepared by	Shaun Gillespie - Director, Gardener
Purpose	To provide employees with a clear schedule for the week. Shows with jobs were successfully completed, and which tasks need re-scheduling.
Size	A4
Number of parts	1
How often?	Created once a week.
How many?	1
Medium	Paper
How long is it kept?	2 years

Data	***Type***	***Size/Example***	***Occurrence***	***Range***	***Source***
Customer Name	String	Ashworth	*Many	N/A	Written by Shaun
Day of Booking	String	Monday	5	N/A	Written by Shaun
Date of Booking	Int	19	5	1 - 31	Written by Shaun
Time of Booking	Real	8 - 9	*	8 - 18	Written by Shaun
Job Completed?	Bool	/	*	Yes / No	Marked by Shaun
‘Overflow’ Customers	String	Mollington	Many	N/A	Written by Shaun

Observation

As an employee of the company, I already had a pretty in-depth understanding of the system they used. However, in order to better understand how the system works, as well as to further identify the system’s strengths and weaknesses, I observed the director, Shaun Gillespie, for a few hours during the organising process in the office, on a Sunday.

During the Sunday, Shaun had to create the timetable for the next week. He started by using a generic template of regular customers, which he had to edit to account for customers who were away or unavailable that week. He then got the list of irregular (‘one-off’) jobs, and slotted them into free spaces based on their location (for

example, he put Paddy, a Borras customer, into a free space alongside the regular Borras customers). This system worked well, until a Marford customer needed to be added, but there were no spaces near the other Marford customers.This meant large amounts of customers needed to be shuffled around in the timetable to create room for the new Marford customer. This was fairly time consuming, and therefore inefficient. Once the timetable was complete, Shaun then noted the ‘large’ jobs which required the assistance of James, and/or Jack. He then proceeded to manually create a separate timetable for each of them, but only writing the jobs they were required for.

As this was the same timetable, just with some items missing, rewriting it for each employee seemed very inefficient to me, and a computerised system would be able to fix this by simply printing the 3 different timetables in seconds. I also noted that during this whole process there was a risk of human error. For example, at one point the same customer had been put down twice for the same job, taking up precious room in the timetable.

I then observed as Shaun calculated some quotes for customers. He wrote down all required materials (i.e 2x 1ft*1ft*6ft timber beam), specifying both the name of the item, and its quantity. It is worth noting that the extreme majority of the time, the materials required are the same as past jobs, and so Shaun has a list of materials and their prices that he uses to then calculate the overall price of each material (i.e 2x timber at £10 = £20). However, there is a small chance the material will not be on that list (for example, Shaun needed a lightswitch to replace an old, damaged one). This means that for the computerised system, there needs to be an option to add new materials to the material list. However, as some items are so niche and unlikely to ever be needed again in a future job, not all new items need to be added to the materials database, and instead the quote should have the option to add a material and price to this job, without having to add that material to the database, where it will then be stored for no reason, taking up storage space in the system.

Another minor thing I noticed is that certain names are abbreviated (for example “Holy Trinity Church” is just written as “H.T.”) in order to save Shan time creating the timetable. As well as this, Shaun does not write down the customer’s address, only their name. This is not a problem, as all of the employees know the customers and where they live, so can go there fine. However, if a new employee was hired, they would have to manually check the customer list for the address of the customer every time, until they learnt it. This problem will never be a win, win with a paper system, as either Shaun loses by spending large amounts of time writing down addresses, or the employee loses by spending large amounts of time checking customer information. However, on a computer system, the program can find the customer’s address from their file, and easily display it in the timetable, fixing both problems without costing Shaun or the employee any time.

Once again this system was highly prone to errors as not only was data being read and written by Shaun, but also processed as he had to calculate the net material price, and the total quote price manually. As well as this, having to manually write down the materials and price per piece (for example, “6ft Fence Panel, £15 each”) was very inefficient as this information is already written down in the materials list, and so a computer copying this information after a human choses it from a dropdown-list or search bar, would save time over a human manually rewriting it.

As an employee of the company, I have had firsthand experience of using the current system, and I have had to add customers and irregulars to the documents. One inefficiency I have noticed whilst doing this is that on the irregularities list when a customer is added, all of their details need to be added (such as address, etc.). However, if the customer is a regular who wants an irregular job, then this is unnecessary, as their information is already stored with the regular customers. As these sets of information are stored on separate documents, it is too slow and inefficient to key information from one to the other, and so it is necessary to rewrite the customer’s information. I believe that the computerised system would solve this as it would be easy and efficient to key data from different files.

From speaking to the employees during this observation, I found out that data entry can take as long as a minute to perform, and that finding a specific item in the files takes an average of 2 minutes, but could take up to 5 sometimes. These times are the targets for the digital solution to beat, as one of the goals of this software will be to reduce the time it takes to perform these regular daily tasks, to save the employees time and maximize their efficiency.

Questionnaire

To get a broad view of the employees’ opinions on the current system, as well as any suggestions for the new system, I planned to create a questionnaire. This would allow me to quickly, efficiently, and cheaply obtain large amounts of information from a large sample of people that could be analysed to pinpoint areas of strengths and weaknesses in the system. However, Carrick Technical Solutions Ltd. only has three employees, and so the use of questionnaires would not be overly useful, as they are only beneficial when used with a large data sample.

Also, questionnaires are best suited to gather quantitative data, and when dealing with qualitative data, often suffer from an inability to probe responses, due to being structured instruments. I am looking for qualitative results, as I would like fairly in-depth opinions on the matter.

Due to these issues, I elected to not use questionnaires, and instead relied on the in-depth interview I had with Shaun, as well information gathered from a few brief conversations with over employees.

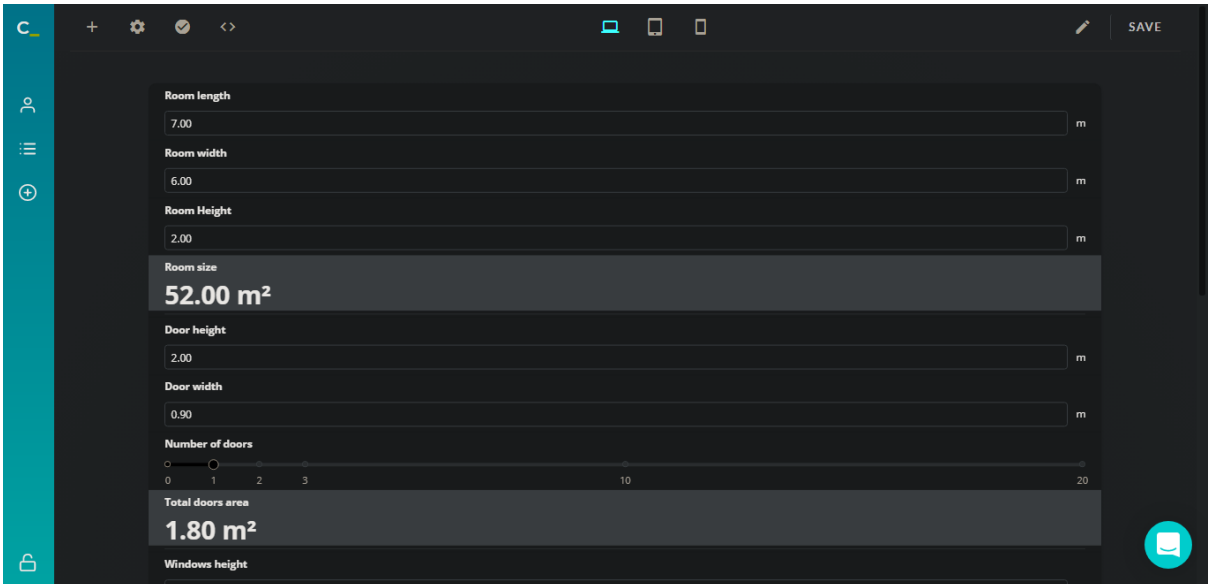
Desk Based Research

In order to increase my understanding of what my system should provide, I chose to look at already existing similar solutions provided by other companies and organisations. By researching these solutions I believe that I can learn from their weaknesses and mistakes, as well as learning from their strengths and ideas that I had not considered.

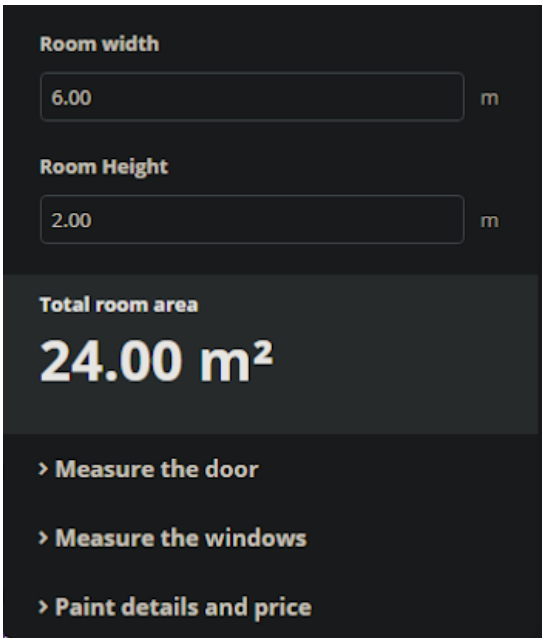
Calconic

The first solution I looked at was a quote calculator by calconic.com. Calconic is a calculation service that offers a broad variety of calculators. They have had years of research and updates to improve their product, and so are very well polished, and such have a lot of strengths that I can learn from.

The system I researched calculated the cost of painting a room, taking several factors such as wall and door area. This calculator is a lot more specific than the general one I plan to make, however it did give me some insight nonetheless.



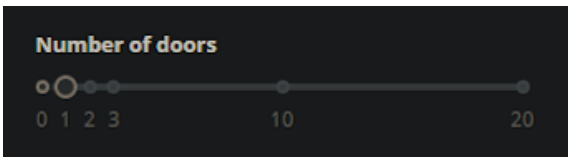
The design of this system is very sleek, and user friendly. The input boxes are very clear with their own, easy to read headings, with larger bold totals that stand out for each section. This makes the calculator very intuitive and easy to use.



The system also has each section as collapsible groups, so that the user is not overwhelmed with an excessive number of input boxes.

Although not evident in the screenshots, the system validates data types as they are input. A field designed for numerical values (such as ‘room width’) not only doesn’t allow for the user to calculate using string values entered, but it does not allow for the user to even input any data other than that of the correct data type. For example, if the ‘a’ key is pressed whilst the ‘room height’ input

is selected, the system simply does not update the field and leaves its contents unchanged.



This field, ‘number of doors’, has a set upper and lower limit (0 - 20), and does not allow the user to input values outside of this range, by forcing the user to select a number from a predetermined scale, and not using a text box.

One issue with this system is that it requires a large amount of data to be input. For example, it calculates the area of the wall by multiplying the dimensions of the wall, with the area of the windows and doors removed. Whilst this is highly accurate, it could be reduced to one input by asking the user for the already-calculated area.

Another issue is that the calculator is set to the unit of square meters (m^2), which cannot be changed. In my system, I will allow the user to choose between the imperial units (such as ft and inches, as DIY supplies tend to be priced in the older imperial system) to better suit employees who favour the imperial system of units. The program can then automatically convert the input data into the correct unit system for its calculations, relatively easily.

System 1 : Calconic, https://www.calconic.com/calculator-widgets/paint-calculator		
Data	**Objective**	**Validation**, or other features
Input features: Web form with text boxes and sliders, broken down into drop down menus. **Data**: Dimensions of surface, area of surface, dimensions of non-surface, area of non-surface, total area of surface, paint efficiency, amount of paint, price of paint (per litre), total price. **Output features**: area of surface, area of non-surface, total area of surface, amount of paint, total price.	Calculate the value for a particular product being ordered.	- Cannot enter string characters into numerical text boxes. - Some numerical values are entered via a slider. - Dropdown boxes

Conclusion:

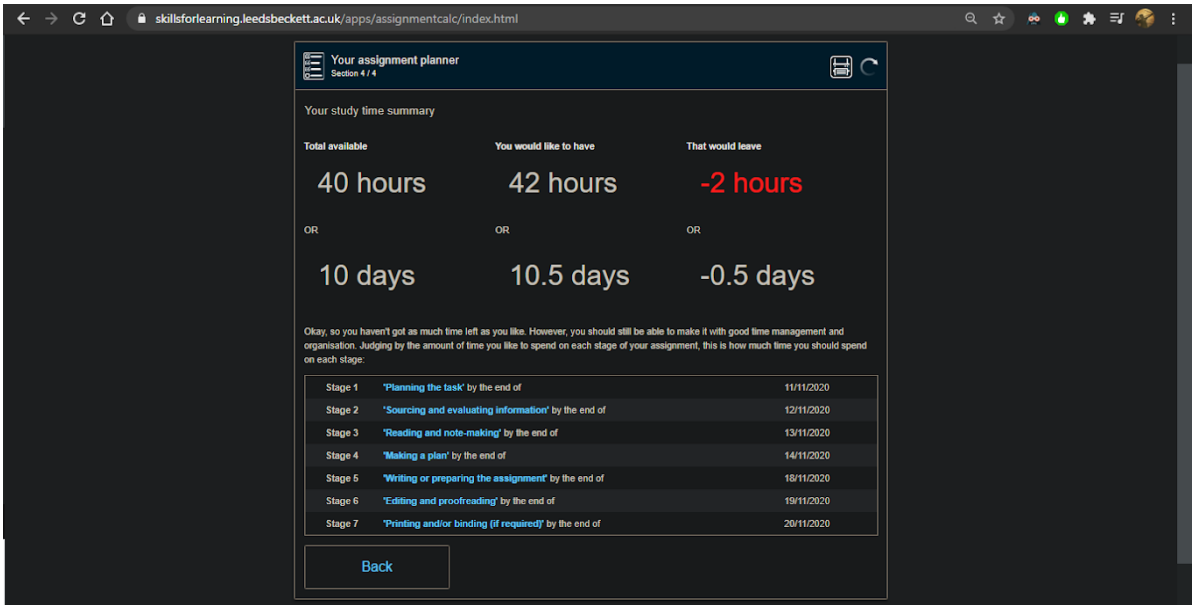
This system works very well for its specific task. For use in my program, it will need to be more general-use and flexible. However, it has many useful features that I can integrate into my system. The validation used by the system is very beneficial and useful, as it prevents the calculations being broken by any incorrect data that could be entered. The use of sliders (or dropdowns, which are similar) prevents incorrect data being entered as the options are predefined with upper and lower limits.

The dropdown boxes are very useful, as it can be used to prevent the user from being overwhelmed with an excessive amount of data input sections, as whole sections can be hidden when not in use. The system also has the calculated data (such as the price) in bold, and larger than the regular input data so that it stands out to the user, making the system easier to use.

Another issue with the system is that it exclusively uses the metric system, whereas in this industry the imperial system is still very dominant, and some employees would prefer this option, and so I plan to improve upon this flaw by allowing the user to change their input unit.

SkillsForLearning

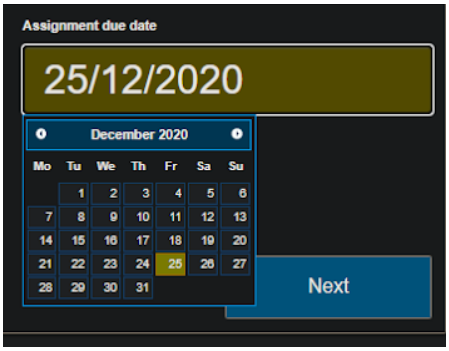
The second system I investigated was a timetable generation software from skillsforlearning. This system only books a single task, in order to help students manage their time for educational tasks and projects. However, the task is divided up into multiple subtasks, which act in the same way the bookings would in my system.



The first thing I noticed after using this system was that the timetable it generates is not visually pleasing, or easy, to use as there are no graphics or

visualisations: only plain text dates. I hope to learn from this, and generate an actual visual timetable for my system.

This software does not organise the subtasks, and instead orders them in the set order of which they are entered, and even if it were, this would not be overly observable from the front-end, as it is all handled in the back-end, and so I would not be able to observe how the system handled this, and so I would not learn much.



This system makes entering dates particularly easy for the user, as it generates an interactable calendar for the user, instead of relying on them to type the correct format, or to select the day, month and year individually from a pre-select list. This calendar also allows the user to know the day of each date, which is often easier for humans to work with than just the numerical values. This also prevents impossible dates (such as February 30th). The system also checks the current date and time, and prevents you from scheduling in the past.

System 2 : SkillsForLearning, https://skillsforlearning.leedsbeckett.ac.uk/apps/assignmentcalc/		
Data	**Objective**	**Validation**, or other features
Input features: Web form with text boxes, numerical sliders and a calendar interface, broken down into multiple menu pages. **Data**: Deadline date, daily time allocation, individual task lengths, available time, required time, spare time, date of each task.	Generate a timetable of tasks (such as *'Planning the task'*, *'Sourcing and evaluating information'*, etc.) based off of factors such as how long each task is, and how much time there is until the project is due.	- Non-existent dates (Feb. 30th) cannot be entered. - Past dates cannot be entered. - Numerical sliders with minimum and maximum limits prevent excessively small or large values from being entered

Output features: available time, required time, spare time, date of each task.		

Conclusion:

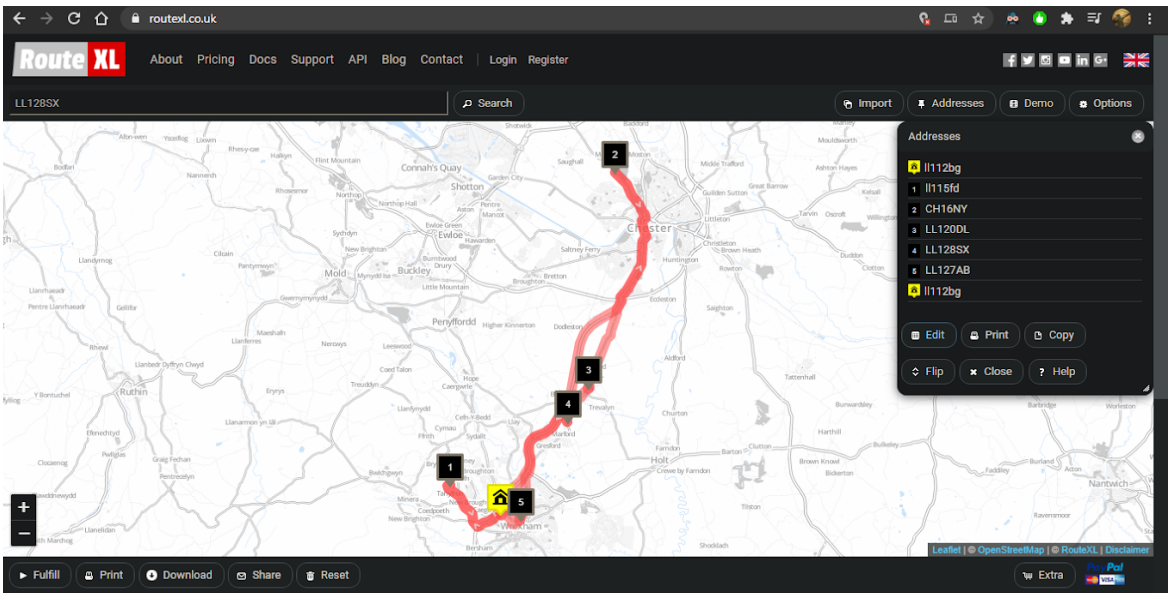
This system also works well for its purpose, by making data easy and simple. The calendar interface is very intuitive and prevents invalid dates from being entered. This system makes this process as easy as possible, and handles all of the calculations.

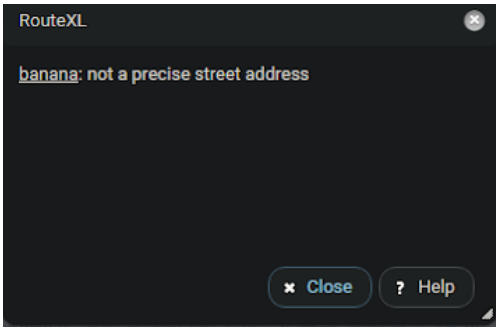
This system suffers from the way its data is output, as it outputs the data as plain text and is quite boring to read, as well as difficult for large quantities of data. I plan to learn from this and output my timetable using a visual timetable which is both easier and nicer to read, and infer data from.

RouteXL

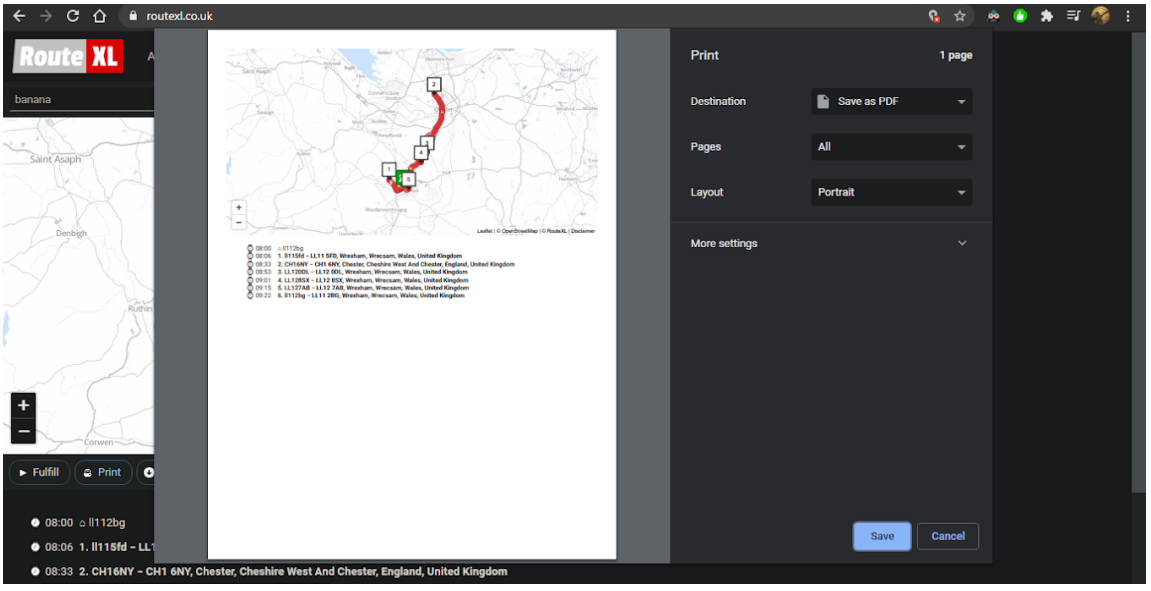
The third system I investigated was a pathfinding software called RouteXL. This software is designed to calculate the shortest way of visiting several user-set stops. For the test I used a number of random addresses and postcodes within the Wrexham and Chester region.

The reason I chose to investigate this system is because when generating the timetable, my system is required to order the customers based off of their location, linking the closest customers. This system uses data from Google Maps to not only locate the precise locations and distances between, but also the exact route which should be navigated. This is too complex for my system, however, and instead the system will use a table of average distances between large areas (such as the distance between Marford and Borras) to calculate this. Although it will be less accurate, it will be a lot more lightweight, significantly easier to program, and will not be internet-dependent.





This system uses validation by checking the formatting of the input ‘address’. For example, when I tried entering ‘banana’ as a location, it detected that it was not a valid address, as it does not follow the “AA000AA” format that postcodes require.



This system also has an option to print the generated route, in which it takes the user to the print screen, with the graphics. This may be slightly too complex to program into my system, however it is a very nice and useful feature, as it will allow users to print off the timetables so that they have a physical copy that they can use with them.

System 3 : RouteXL, https://www.routexl.co.uk/

Data	**Objective**	**Validation**, or other features
Input features: multiple addresses (postcodes) **Data**: distance between the addresses **Output features**: route, list of ordered addresses (following route)	Generate a route between different addresses to minimise the distance of the route.	- Postcodes must match the AA000AA format, otherwise they are not accepted. - Print option

Conclusion:

This system does an admirable job at making the process as easy as possible for the user. The system disallows non-address data from being entered, meaning there is nothing that can break the calculations. This system is online as uses the Google Maps API to function, however a very similar system can be created without this by using a predefined list of different areas (Gresford, Marford, Borras, etc.) and the distance between them.

This system creates an exact route between the points, which is not necessary for my system, as it will be incredibly complex and complicated to create, and all that is required for my system is to create a list order.

One nice feature the system has is the ability to print out the results directly, without having to screenshot the data. This feature could be complicated to create and implement, however it will be an invaluable strength of the system, and would be very useful for the users.

Stakeholder Investment

Shaun Gillespie, Director of CTS Ltd.

- Shaun uses the current systems to schedule the weekly timetables and to calculate quotes for customers.
- Shaun also uses these documents to create a projected income (earnings, not profit) for the month, and the remainder of the year, and will often compare these projections to past months'/years' earnings.

- Shaun has to manually read through a list of customers, perform calculations and write down the information. As this is all done manually, this allows for human-error in three different areas (reading data, processing data, writing data), whereas if the system was computerised, the only area for human error would be reduced to inputting data. This would increase efficiency and benefit Shaun.
- The automated system would not only increase Shaun's efficiency by reducing errors. It would also become a time-saving tool as it will be able to calculate and generate timetables significantly faster, and possibly more efficiently, than a human. This time-saving benefit would also apply to calculating quotes, and comparing earnings.

James Gillespie, Employee
- James uses the current system a lot less than Shaun, and only uses the booking part of it, to look at the timetable, and to possibly add new customers, or irregular jobs to the system.
- James would also be aided by the system reducing the opportunity for human errors. As James does not process any data, only inputs or outputs it, he would not be affected by the time-saving aspect. However, he would be affected by the system if the computer generates a more time-efficient timetable, that reduces transportation time more efficiently than the current system.

Jack Gillespie, Employee
- Jack uses the current system the same as James, to look at the timetable, and to possibly add new customers, or irregular jobs to the system.
- Jack would also benefit from the system reducing the opportunity for human errors and creating more efficient timetables. Jack would also not be affected by the time-saving aspect, as he does not have to process data.

Inputs, Processes, and Outputs

From the data collected in my interview, document analysis, observation and desk based research I have noted the different inputs, processes and outputs that will be used by the system. The inputs required for this system are:
- Customer details (name, address, etc.)
- Material details (type of material, quantity of material, cost of material per unit)

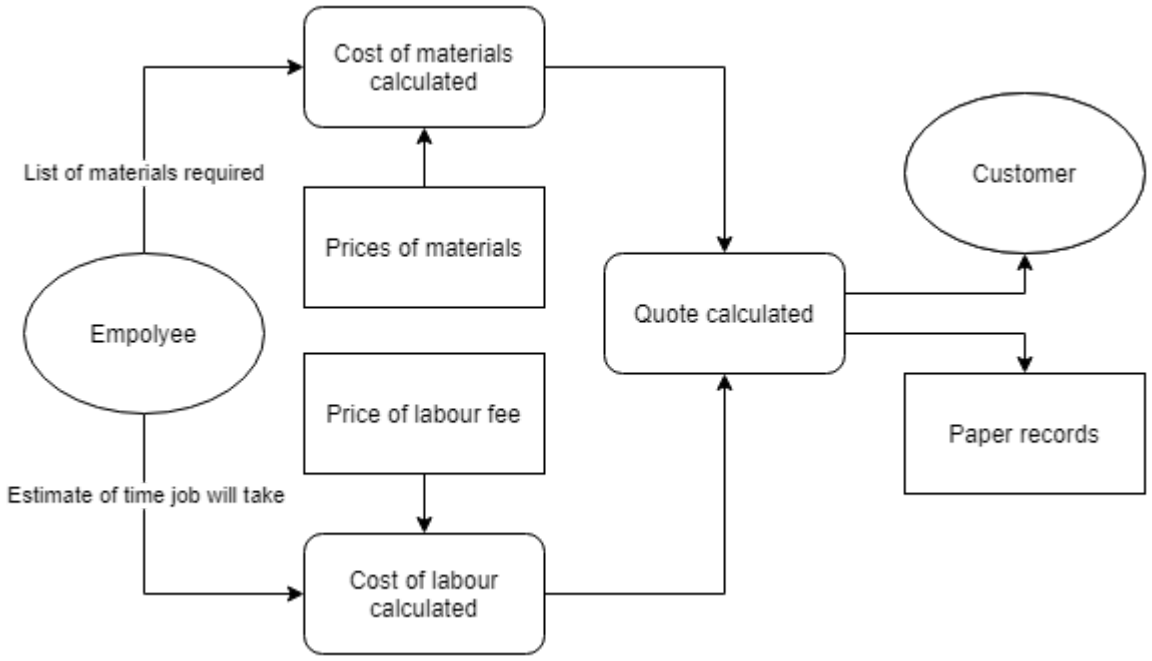
The processes are:
- Generating the timetable of customers
- Calculate the subtotals, and totals of quotes
- Calculate projected earnings
- Calculate the difference between projected earnings, and past earnings

The outputs are:
- The timetable, along with customer details

- Projected earnings
- Past earnings
- Quote subtotals and totals
- Graphs (bar, possibly line) showing data in respect to time, for example the earnings of the company over a user-defined time period. This graph can be designed to display multiple sets of data simultaneously for comparison purposes.

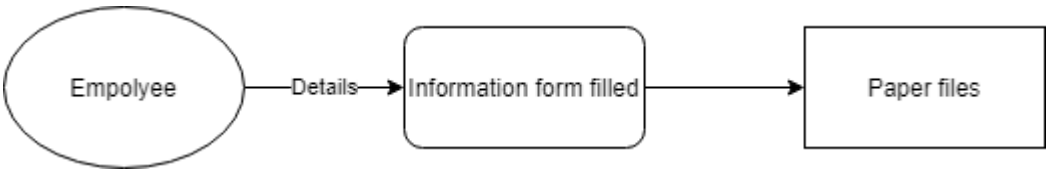
Data Flow Diagram

Creating a quote.



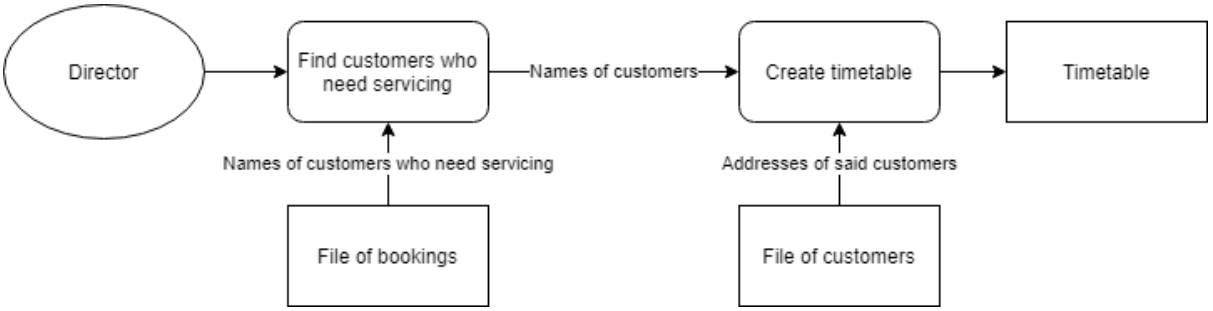
This diagram shows that an employee can calculate the price of a quote, by first creating the labour fee, using the time the job will take, and the fixed hourly rate for the job. The employee will also calculate the material cost, using a list they have made of materials required, and the prices of materials that can be found from the company's paper records. The employee will then summate these prices to calculate the total price of the quote, which can then be given to the customer and/or stored in the company's paper records

Adding a customer, booking, material, or other data to the company's records.



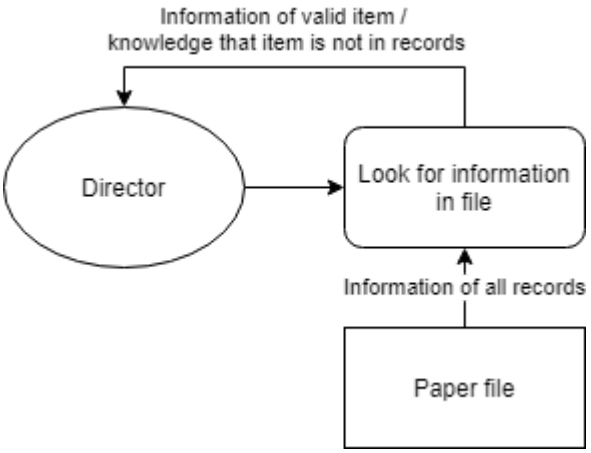
The employee can store the information about a customer, booking, etc., by entering the information into a form, and then storing that form in the paper file.

Creating a timetable of customers for the week.



This diagram shows that the director creates the timetable by getting the customers that require work doing from the booking files, and then looking at those customers' addresses by cross-referencing the names with the customer files, and then arrange them in such a way that they are collated and ordered by area.

Finding details of a customer from the paper records.



This diagram shows that the director can access information by looking through the files.

Limitations of the Existing System

Throughout my investigation, the imitations of the current system have become apparent to me, particularly from my observation.

One of the largest issues I have identified, by far, with the system is the opportunity for human error in the handling of reading, calculating, and writing data. Whilst no computer system can truly resolve this, it can significantly reduce the likelihood of it happening, by handling all of the calculations, and by using validation to ensure data is entered correctly. Similar to this is that in order to improve time-efficiency, a lot of data is abbreviated or excluded. This saves some time, however it can create problems as other employees may struggle to understand some of the abbreviations.

This limitation affects all employees, and so it is vital that the computer system addresses this.

Another major issue with the paper system is how time-consuming and inefficient some of the processes are. For example, creating the timetables and quotes can be very time consuming for the employees involved. However, a computer should be able to significantly speed up the time these processes take.

If both of these issues are addressed, then the new system will be both more time efficient and more accurate. These are simple solutions, however will be greatly beneficial for the company.

Whilst not as major, there are also issues regarding the loss of data, as paper copies could be lost, misplaced or damaged. Whilst a computer system is not immune to data loss (as the files could become corrupted, or deleted) it is possible to automate a backup system, which can minimise the impact of these losses. Similar to this, data could even be stolen fairly easily if it is stored on paper, and so confidential customer information could be taken. A computer system greatly reduces the chances and danger of this happening, as sensitive data can be encrypted and locked safely behind a password, meaning that even if the data is stolen, no sensitive information can be extracted from it. Computer systems can often increase the risk of theft due to hacking, however as this system will be on an offline computer, there is no way for hackers to access the system remotely.

Working Specification

From all my research into the current system I have created a working specification that has highlighted the following are important for the new system.

The first screen the user will use will be the login screen. This menu will prevent any unauthorised personnel from accessing the system, and the sensitive information that is stored in the system. This screen will require both a username and password to be entered, and a button to proceed to the next screen if the information is correct. The screen the user will proceed to will depend on the access-level of the entered details. This system will be secure as the usernames and passwords will be encrypted, making near-impossible to access them without the system. This will make the system secure, and prevent theft of data. If the details are not correct, then the user will not advance, and will stay on the same screen until valid credentials are entered. Perhaps if the user fails to enter correct details too many times in too short a space, then the system will lock to prevent brute-forcing.

The system will have a central menu, that will then allow the user to navigate to the next menu of their choosing, such as the quote calculator, the booking system, the timetable system, etc. Which menus are available will depend on the access-level of

the user. For example, only the quote calculator and financial aspect of the system will be available to the director, with standard employees only being able to access the booking and timetable menus.

The booking screen will be used by all employees. This screen will be composed mostly of text boxes to input the required information, such as customer name, job type, job length, address (area), postcode, price of job, etc, and a bool field of whether this is to be a regular (recurring) job or an irregular (‘one-off’) one. All of this data will be stored into the files after the user presses a submit button. Before the system saves this data, it will verify some of the fields -- for example the numerical values must be checked to be numerical data types, to prevent them from breaking the calculations, and certain fields must match a specific format, such as the AA000AAA format of postcodes, as well as presence checks to ensure crucial fields are not left blank. If any of these verifications fail, the system will not submit the data, and will highlight the issues to the user. Once all of the data passes the verifications successfully, the data can then be stored into the system for future use. Once a new job is added, the system will also have to update the timetable and add the new job.

This update could be done when the timetable menu is opened instead, however in theory the timetable will be opened multiple times per day, whereas new booking would be added much less frequently, meaning that the timetable generation calculation has to be performed a lot less.

The timetable menu will simply fetch the already-generated timetable and output it to the user. This will be done visually, using graphics to create a standard timetable format. The system will not automatically display all bookings however. It will only display the jobs on the timetable that are for the specific user that is logged in. This section will have the option to print out the timetable as hard copies, as physical copies will be needed for employees to take with them, and the computerised system will only be available on an offline standalone computer, which will not be accessible out of the office.

The quotation calculator will be a relatively simple form, which will consist of input text boxes, and possibly sliders or other input methods. Similarly to the booking form, validation checks will be performed on numerical values to ensure all calculations work smoothly. Instead of validating data at the end after submission, I may make the system validate the data as it is input, similarly to the Calconic system that was researched in the desk based research. Data such as the price-per-unit of each material will be stored in a database file, which this form will have to read to use in the calculation. Another thing is that unlike the Calconic system, I will allow the user to select between the metric and imperial units for each measurement (which will then be converted into the appropriate unit for the calculations). The layout of this form will be very similar to the Calconic system, using bold and larger fonts to

highlight calculated data as opposed to the input data, and could benefit from dropdown menus.

Another menu will be the financial records form. This form will allow the user to go through the earnings of past months and years, sorting the data in different ways, such as seeing which type of jobs create the greatest amount of money. This system will be available only for the director, as standard employees should not be able to access this confidential financial information. This system will be input-less, only using already stored data and calculations in order to output useful information to the user.

Technical Explanation

In this project, I will be using text files to store the system's data instead of databases. I have chosen to do this for a multitude of reasons, primarily that text files use less memory so are a more efficient storage method, as well as the fact that using my own system and code gives me complete control over the storage system, and can tailor it to my specific needs, to increase the efficiency of the program. The files will however be normalised to avoid unnecessary duplication and inconsistencies. The project will store the data across several files, so as to avoid confusion when accessing data, as only using one file, or using a new file for each item of data, would be inefficient and overly complex and confusing to handle. Data will be inputted and stored via the system, and only once the system has validated (and verified, in the case of passwords, for example) it and ensured that there are no errors that will break any calculations. The files will be fixed length records in this project, as they are simpler to develop, and have very few drawbacks.

The system will have to use forms applications in order to allow the user to add and edit the records stored in the data files, as the file handling will be done by the system. This could be done via command line, however this is not at all user-friendly, especially for non-programmers. When the system reads data from the files, it should be displayed in non-editable form items such as labels as opposed to text boxes. This is to prevent the data being overwritten accidentally by the user. When overwriting or deleting data, a confirmation system will be required to ensure that users do not mistakenly remove essential information. A backup system can be automated, to reduce the chances of data loss or corruption affecting the system as much, or as significantly.

The majority of data entered via text boxes will require validation to ensure that the data is usable by the problem without causing any issues. Almost all text boxes will need presence checks to make sure that blank data isn't stored. Numerical values will require data type checks to make sure they are not in the string form, as if they are they will not be usable in calculations and will break the program. Length checks will also be important as the files will only have a certain amount of characters

available per character, and if the item is larger than this limit it will cause problems. The reason I have chosen to use fixed length records, and not variable length records, is that they are easier to program and use, and it is also easier to estimate the size of each file.

The files will be stored on the computer's hard drive (or solid state drive, if it has one).

The majority of variables and objects will be local, so as to avoid accidentally overwriting or altering them by confusing all of the variables. If variables are local, they can only be edited by their respective code, and so it is a lot harder to confuse variables. Some variables may benefit from being global, so that they can be accessed from across the system -- however public get and set methods can work around this, and so global variables can be avoided altogether if necessary.

A variety of data types will be required. Strings will be required for texts such as names, addresses, etc. as this is basically the only way to store them. Numerical values such as quantities and prices will be stored as integers or real numbers. This is necessary as only these values can be used in mathematical calculations, whereas strings cannot. However, strings will be used to input these variables as part of the validation process. The numbers will be stored as strings, which will then be converted into the correct numerical value, and if the program cannot complete this conversion it will know that an incorrect value has been entered. Some values will also need to be stored as booleans (true/false and yes/no options) such as whether a customer is a regular or not.

This project will be using classes, instead of one large procedural script, as this will remove the need to rewrite code, as data from anywhere in the system can be fed the needed classes/objects in order to process it. Classes also make the program easier to test and debug, as only the one algorithm needs testing, not multiple versions of the same algorithm. Using classes also means the classes can be tested individually from one another, instead of having to test the entire program at once, as you would if it was written as a single algorithmic script. Classes will be used to validate data, to prevent the same function being re-written in multiple places in the project. This reduces the required testing, as well as making cleaner, more storage-efficient code. Encryption will also be a separate class, as it will be required by multiple forms (for example both the login system and user creation systems will need this) and so will be in its own class, and called by these forms when needed.

Volatile memory, such as arrays, will be used to temporarily store data before storing it into the files, and to temporarily hold the data that has been read before it has been processed.

Graphs will be used to output data in an easy to understand visual format, for example, displaying the company's earnings over a given period of time. These graphs will be able to handle multiple periods of time simultaneously, allowing the user to directly compare data, as well as just being shown data, making the graphs a useful tool. The calculations for these graphs will be within the graph's form, as only one form will be outputting graphs, this code will not need to be accessed from anywhere else in the program, and so does not need to be in its own individual class. This code will however be broken down into functions, creating a modular system that can easily be expanded with new additions, and not requiring code to be written for new implementations to work.

As all data will be unsorted, linear searches will be used. This is not the most time or memory efficient search method, however as there will only be a relatively small amount of data used in this system, as CTS is a small company of 3 employees and around 70 customers, all search methods will be very similar in performance results with an unnoticeable difference to the user in speed. As all these search methods will be very similar performance-wise, I have chosen to use linear searches as they can be performed on unsorted data, and so removes the need for sorting algorithms, as well as being very simple and easy to develop.

Some data will have to be sorted as part of the timetable generation process. Once again only a relatively small amount of data will need to be sorted, and so the time differences of different sorting algorithms will be negligible. As this is the case, I will use bubblesort algorithms when sorted is required, as once again it is an easy and simple sorting algorithm.

Objectives

Quantitative Objectives

1. It should be significantly faster to generate a timetable.
2. It should be faster to find a customer's details.
3. Projected earnings for the week should be automatically generated.
4. Information should be automatically checked for errors, such as words for numerical values.
5. All staff should be able to access the required, and only the required, features and functions of the system.
6. The director must be able to create, manage and delete user accounts.
7. Users should be able to add entries and information easier.
8. All users must have securely encrypted usernames and passwords.

Input Objectives

9. **Login Screen**. Username and password should be typed using a keyboard, and submitted using a mouse to interact with a button.
10. **Navigation Menu**. The user should navigate to the required menu using the mouse to select the corresponding button.
11. **New Regular Booking**. The customer's details (name, address, postcode, etc.) should be entered using a keyboard. Some details, such as 'type of job' should be selected from a dropdown menu using the mouse.
12. **Financial Menu**. The director should be able to select dates and filters through the use of the mouse.

Validation of Input

13. **Presence checks** should be used on almost all data inputs (barring optional inputs, and inputs where there is a default value, so it cannot be blank).
14. **Data type checks** should be used on all numerical values that are needed for calculations, such as quantities, prices, etc,
15. **Format checks** should be used on patterned values, for example postcodes should match the AA000AA format, and phone numbers should match the 07xxxxxxxx format.
16. **Length checks** should be used on all file-stored inputs, to ensure that the entries fit into the records.
17. **Two-pass verifications** should be used when creating new users, to ensure that the correct passwords are entered.
18. **Exclusive checks** should be carried out when new users are created to ensure that the username in question is not already taken.

Processing Objectives

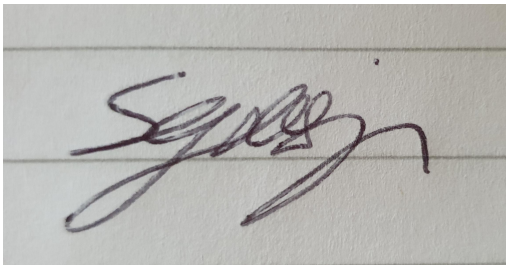
19. Timetable should be generated to ensure that customers are ordered based on their geographical proximity.
20. Quote prices should be calculated from the input data.
21. New user data should be encrypted.
22. Projected earnings from the week's jobs should be calculated.
23. Past earnings should be comparable and searchable.

Output Objectives

24. Timetable should be viewable by all staff, as a visual representation.
25. Timetable should be printable.
26. Quotes should be viewable by the director. Results of calculations should be viewable by the director.
27. A list of users should be viewable by the director.

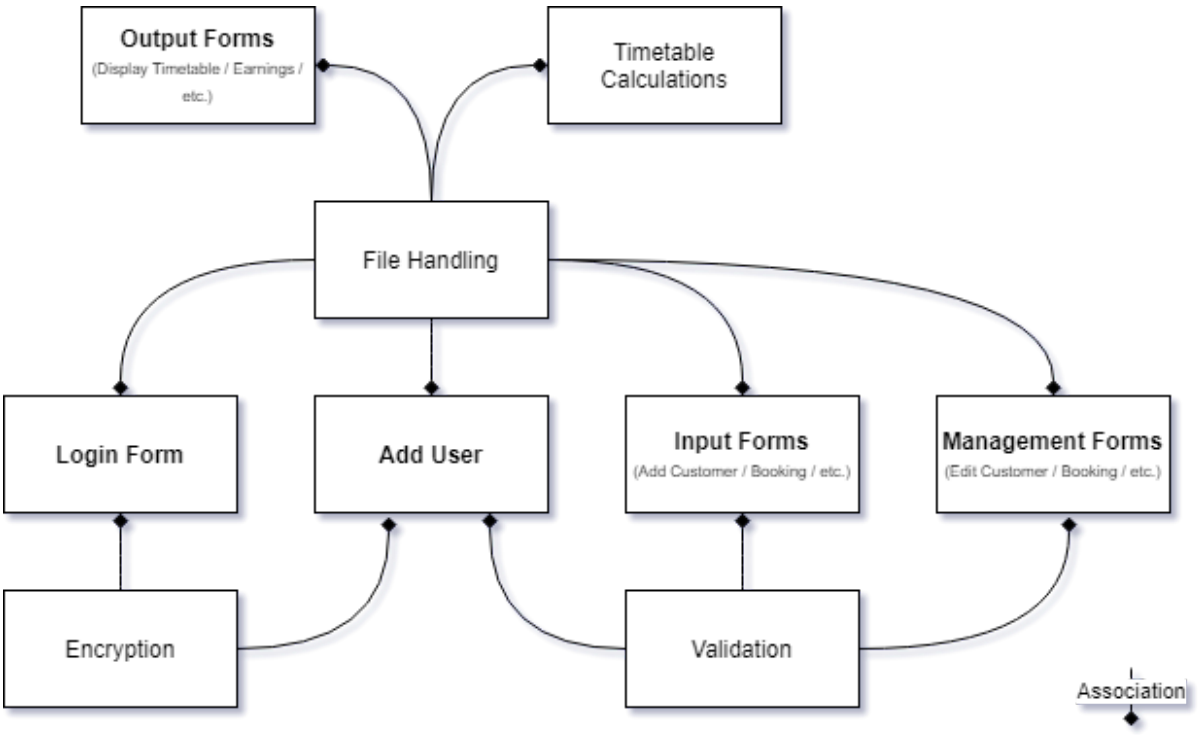
28. A graph should display the earnings of days between two sets of dates, where they can be directly compared.

Signed and approved by ***S. Gillespie***



Design

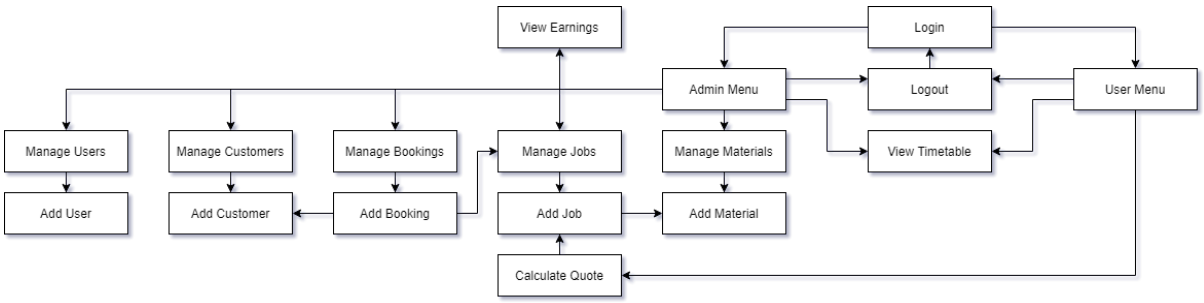
Unified Modeling Language (UML)



The above UML diagram shows how the forms and classes of my system will interact with one another. A file handling class will be used for the input forms, to store the new data into non-volatile storage, as well as for the output forms, to fetch that data to be displayed to the user. The file handling class will also be used by some processes, such as the timetable calculations, as it will need to fetch data for it to work with, as well as storing the result. The login form (as well as the add user form) will make use of an encryption class, as well as the file handling class, as for security purposes, the usernames and passwords of users will need to be encrypted. Several classes will also make use of a validation class, which will handle all validation techniques for the program.

This class will be used by all data input forms, to ensure the correct data is being inputted into the system, to ensure that no runtime errors will occur as a result of it.

Tree Diagram



Hierarchical chart showing navigation between forms. (Admin Menu / User Men are displayed separately, however will be the same form, with alterations for different access levels. Same for Manage Users / Customers / etc.)

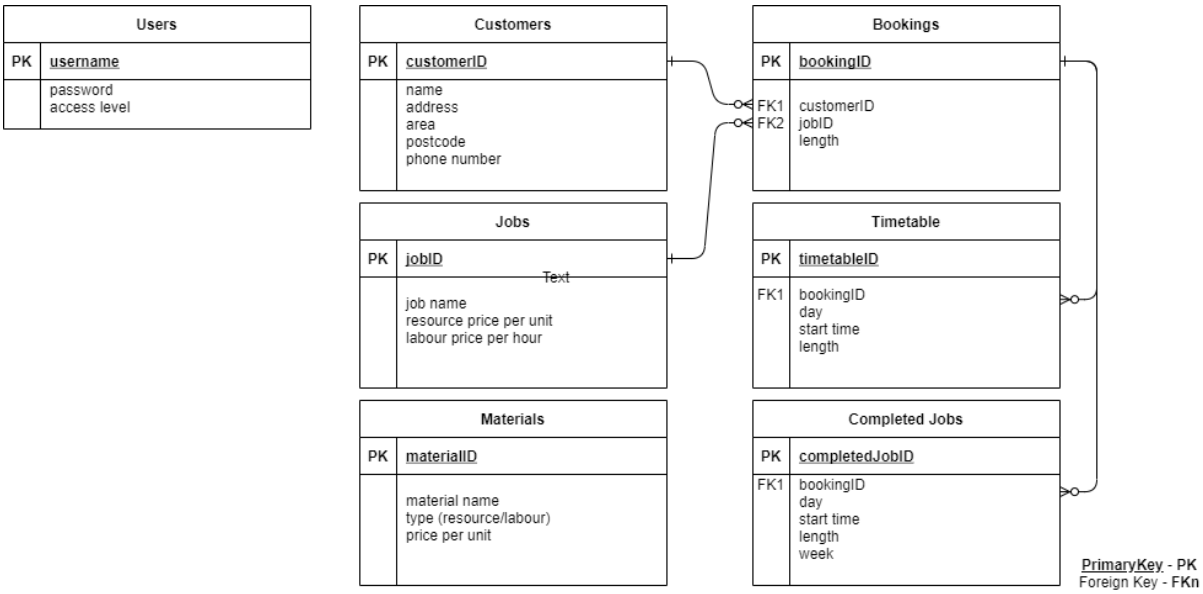
Objectives referenced to Forms

Objective		**Form Relation**	**Link**
2.	It should be faster to find a customer's details.	View Customers.	Users will be able to view all of the customer's data in one place, just by searching for them.
4. (& 14 - 19)	Information should be automatically checked for errors, such as words for numerical values.	All data input forms (Add User, Add Customer, etc.)	These forms will all make use of a validation class, to help reduce the chances of incorrect data being input, by ensuring the data is of the correct type, etc.
5.	All staff should be able to access the required, and only the required, features and functions of the system.	Menu form. (/ Login Form)	The menu form will provide the user with access to all of the required forms. The data given on the login form will determine whether or not the user is an admin or not, and then will give the user the correct menu for them, to

			prevent non-admins having access to admin features.
6. 28.	The director must be able to create, manage and delete user accounts. A list of users should be viewable by the director.	Manage Users. Add User.	The manage users form will allow admins to view the list of existing users, alter their access level, and delete them. The add user form will allow admins to create new users.
7.	Users should be able to add entries and information easier.	Add Customer. Add Booking.	Users will be able to input the data into these forms. When adding bookings, the system will aid them by automatically copying the customer's secondary data (address, phone number, etc.) into the booking after the user has selected them via name. This prevents the user having to manually find that data and input it themselves.
8.	All users must have securely encrypted usernames and passwords.	Login. Add User.	These forms will both make use of an encryption class that will encrypt the username and passwords of users, to prevent people from being able to access the system simply by checking the users.txt file.
9 / 11 / 12.	[Data] should be typed using a keyboard, and submitted using a mouse to interact with a button.	Login. Add Booking.	The user will be able to input the username and passwords data via keyboard into the form's text boxes, and will be able to interact with the form, by selecting the 'Login' button, which will trigger the data comparison.
10.	The user should navigate to the required menu using the mouse to select the corresponding button.	Menu.	The menu form will have buttons on it, allowing for the user to interact with the form, and be directing to their desired form via the mouse, by moving to, and selecting the corresponding button.
25.	Timetable should be viewable by all staff, as a visual representation.	View Timetable.	The form will create a visual timetable (using buttons) to represent the timetable data in an easily understandable way

			for users. The buttons will have their item's data (customer name, address, etc.) on them as their text. Pressing the button will display the whole range of information.
26.	Timetable should be printable.	View Timetable	The timetable form will have a button to print off a physical copy of the timetable.
27.	Quotes should be viewable by the director. Results of calculations should be viewable by the director.	Manage Jobs. Calculate Quote.	The director will be able to view all previously generated and saved quotes via the manage jobs form (using the view jobs function).
28.	A graph should display the earnings of days between two sets of dates, where they can be directly compared.	Graph	The director will be able to generate a graph that displays the earnings from completed jobs, by calculating the values of each day's jobs as a summation of their component prices.

Entity Relationship Diagram (ERD)



Above is the entity relationship diagram for my system, and shows all of the files the system will use for storage, as well as their contents, and their links to each other.

The users.txt file is an independent file that will hold the user's data, such as login details and access level, that has no relationship with any other files.

The customers.txt file will contain all of the customers' data, such as name, location, and contact information.

There is also the job.txt file which will store the details, such as name and price, of a job (/quote).

The bookings.txt file will store user-made bookings, and will use foreign keys to link them to the customers and jobs found in customers.txt and jobs.txt. The file will also contain the length of the job.

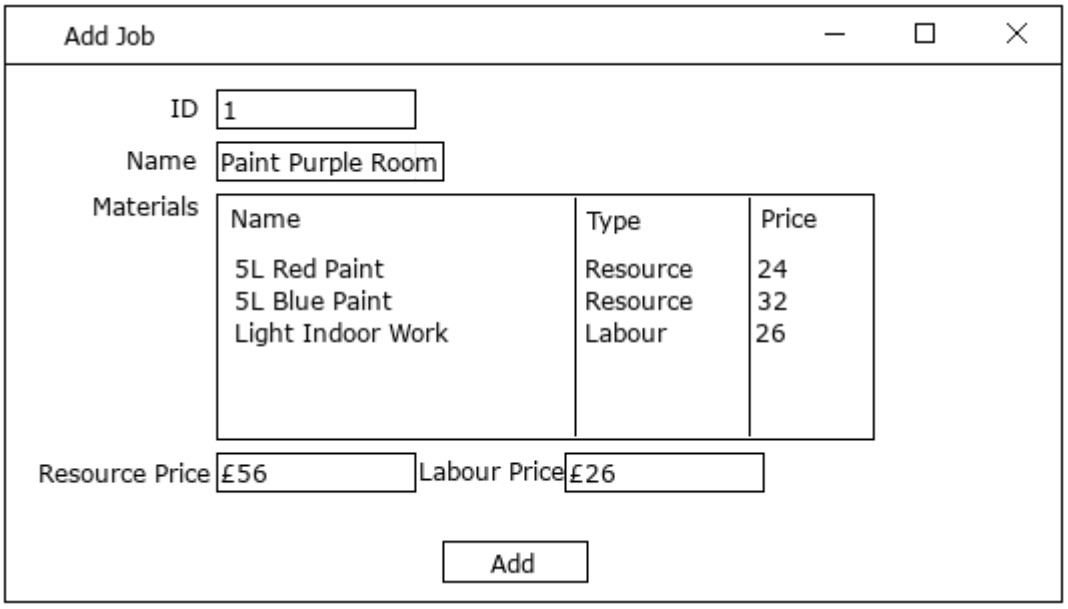
Timetables.txt is a file which stores the foreign ID of its respective booking, as well as the day and start time that the item is on, and the length of the item. The length will likely be the same as that in the bookings.txt file, **however**, it is not a foreign key as sometimes the lengths will not be the same. For example, if booking 1 is 4 hours long, it may have to be split into two timetable items (number 1 and 2) in order to have a lunch break in between. This will mean that the two items will still be the same booking, with the samebookingID, however they will have different lengths (perhaps 2 hours each, for example).

completedJobs.txt is very similar to timetable.txt, except the timetable stores the current timetable, whereas completed jobs stores all past timetable items and so are separate from the current timetable. Completed jobs also have a

week field, which stores the week in which the job happened (the date of that week's Monday is used).

Lastly there is the materials.txt file. This file is used to determine the contents of the jobs.txt file, however it is not directly related to it, as the system takes values from the materials file, sums them, and *then* stores them in the jobs file. Below is an example:

Materials			
ID	Name	Type	Price
1	5L Blue Paint	Resource	32
2	Light Indoor Work	Labour	26
3	5L Red Paint	Resource	24



Jobs			
ID	Name	Resource Price	Labour Price
1	Paint Purple Room	56	26

As shown in these tables, none of the data from materials is directly in the jobs file, as the resource and labour prices and calculated by the form as a summation of the material's contents, and that value is what is stored.

File Descriptions

File Name	users.txt
Information	Stores the data of the system's users, such as login details, for the login

	system. As well as access levels. I will use sequential files to store and retrieve any details or data stored in the file, as these can be read straight into an array of records in memory.
Relations	None

Field Name	**Data Type**	**Fixed Length Position**	**Validation**	**Description**	**Example Data**
username	String	len12 pos1	Length check, presence check	The **primary key** of the file. Is the username used by the user during the login process. The username input during the login process is compared against this value.	Gandhi
password	String	len12 pos13	Length check, presence check	The password input during the login process is compared against this value. *This field is encrypted.*	a#0E5mo3
access level	String	len1 pos25	Assigned value.	The value of this field determined whether the user is an admin or not, as so will determine which navigation menu they are sent to after logging in, which will affect which forms they can access.	1

File Name	customers.txt
Information	Stores the data of the company's clients. I will use sequential files to store and retrieve any details or data stored in the file, as these can be read straight into an array of records in memory.
Relations	Referenced by Bookings

Field Name	**Data Type**	**Fixed Length Position**	**Validation**	**Description**	**Example Data**
customerI	String	len3	Assigned	The **primary key** of the	42

D		pos1	value.	file. Is a unique number used for identification.	
name	String	len25 pos4	Length check, presence check	The name of the customer. This is for the user to identify them easier than having to use the primary key.	Ian Mellon
address	String	len25 pos40	Length check, presence check	The first address line of the customer (number, street name). This is for the user to know where to go.	25 Bloom Avenue
area	String	len12 pos65	Length check, presence check	The area in which the customer lives. This will be used to group the customers for the timetable generation process. It is also used to help the user to know where to go.	Brymbo
postcode	String	len7 pos77	Format check. Must be of the 'AA000AA' format. Length check, presence check	The postcode of the customer. This will be used to group the customers for the timetable generation process. It is also used to help the user to know where to go.	LL115FD
phone number	String	len11 pos29	Format check - must be of 00000000000 format. Data type check - must be numerical. Length check, presence check	The phone number of the customer. Stored exclusively for the sake of the user, allowing them to contact the customer if they need to. Not used by the system in any way.	01978 919 490

File Name	jobs.txt

Information	Stores the quotes. I will use sequential files to store and retrieve any details or data stored in the file, as these can be read straight into an array of records in memory.
Relations	Referenced by Bookings

Field Name	**Data Type**	**Fixed Length Position**	**Validation**	**Description**	**Example Data**
jobID	String	Len3 pos1	Assigned value.	The **primary key** of the file. Is a unique number used for identification.	42
job name	String	length20 pos4	Length check, presence check	This is for the user to identify them easier than having to use the primary key.	Mowing
total resource price per unit	Real	len6 pos23	Data type check. Range check (0 < x < 1000). Must be a positive numerical value, of reasonable value. Assigned value.	The price it costs per unit (litre, metre, square metre, etc.) for the materials. This value is the summation of resource prices of the materials. This is explained using tables above.	0
labour price per unit	Real	len6 pos28	Data type check. Range check (0 < x < 1000). Must be a positive numerical value, of reasonable value. Assigned value.	The price it costs for the labourer(s) to work per unit (hour). This value is the summation of labour prices of the materials. This is explained using tables above.	17

File Name	materials.txt
Information	Stores the different materials and their prices. I will use sequential files to store and retrieve any details or data stored in the file, as these can be read straight into an array of records in memory.
Relations	None

Field Name	**Data Type**	**Fixed Length Position**	**Validation**	**Description**	**Example Data**
materialID	String	len3 pos1	Assigned value.	The **primary key** of the file. Is a unique number used for identification.	42
material name	String	len25 pos4	Length check, presence check	This is for the user to identify them easier than having to use the primary key.	Timbre 75x70x4.2
type (resource / labour)	String	len8 pos29	Selected from dropdown.	This is whether the material is a physical object (e.g. timber) or a service (e.g mowing)	Labour
price per unit	Real	len6 pos37	Data type check. Range check. Length check, presence check	The price of the item. Used in the quote calculations.	20

File Name	bookings.txt
Information	Stores the jobs that need to be performed. I will use sequential files to store and retrieve any details or data stored in the file, as these can be read straight into an array of records in memory.
Relations	Foreign keys linking to customers.txt, and jobs.txt

Field Name	**Data Type**	**Fixed Length Position**	**Validation**	**Description**	**Example Data**

bookingID	String	len3 pos1	Assigned value.	The **primary key** of the file. Is a unique number used for identification.	42
customerID	String	len3 pos4	Assigned value.	A **foreign key** used to identify the customer this booking is for. This is used to find the corresponding area and postcode, for use in the timetable generation procedure.	117
jobID	String	len3 pos7	Assigned value.	A **foreign key** used to identify the job that this booking consists of.	16
job length	Int	len3 pos10	Data type check. Range check. Length check, presence check	This is how long the job will take (in minutes). This is used to fit the items into the timetable.	600

File Name	timetable.txt
Information	Stores the jobs that need to be performed, that have been sorted by location, and sometimes split in length, in order to form the complete timetable. I will use sequential files to store and retrieve any details or data stored in the file, as these can be read straight into an array of records in memory.
Relations	Foreign keys linking tobookings.txt

Field Name	**Data Type**	**Fixed Length Position**	**Validation**	**Description**	**Example Data**
timetableID	String	len3 pos1	Assigned value.	The **primary key** of the file. Is a unique number used for identification.	42
bookingID	String	len3 pos4	Assigned value.	A **foreign key** used to identify the booking this timetable item is for. This is used to identify the corresponding	117

				customer and job, for the user to see, as well as for some monetary calculations.	
day	String	len1 pos8	Assigned value.	A number from 0-4 that represents the day of the week the job is on (0 = Monday, 4 = Friday)	0
start time	Int	len3 pos12	Assigned value.	This is when the job starts: in minutes, after 8 o'clock. 0 = 8am 60 = 9am 90 - 9:30am etc.	120
job length	Int	len3 pos16	Assigned value.	This is how long the job will take (in minutes). This is calculated by the timetable generation process.	330

File Name	completedJobs.txt
Information	Stores the timetable items that have been completed. This is used to keep track of what has been done -- mostly for the sake of invoicing. I will use sequential files to store and retrieve any details or data stored in the file, as these can be read straight into an array of records in memory.
Relations	Foreign keys linking tobookings.txt

Field Name	**Data Type**	**Fixed Length Position**	**Validation**	**Description**	**Example Data**
timetableID	String	len3 pos1	Assigned value.	The **primary key** of the file. Is a unique number used for identification.	42
bookingID	String	len3 pos4	Assigned value.	A **foreign key** used to identify the booking this timetable item is for. This is used to identify the corresponding customer and job, for the user to see, as well	117

				as for some monetary calculations.	
day	String	len1 pos8	Assigned value.	A number from 0-4 that represents the day of the week the job is on (0 = Monday, 4 = Friday)	0
start time	Int	len3 pos12	Assigned value.	This is when the job starts: in minutes, after 8 o'clock. 0 = 8am 60 = 9am 90 - 9:30am etc.	120
job length	Int	len3 pos16	Assigned value.	This is how long the job will take (in minutes). This is calculated by the timetable generation process.	330
week	String		Assigned value.	The date of the Monday of the week the job was completed. This can be used alongside day (week + day = date) to calculate when the task was performed.	22/01/2020

Presence checks are applied to all user input fields (this is done by comparing the field to null, and only accepting as valid if it is not), to ensure that no blank data is input.

Length checks are also applied to all fields, to ensure that no data exceeds the field length position used to store it within the file.

Data that is generated by the system (e.g. the primary keys) should always be correct, and so do not require the validation process. These values are labelled as 'assigned value' in the validation section of the above table.

Validation Routines

Presence Check

```
Private Function PresenceCheck(variable)

        If variable = null 'if variable is blank
                Return False 'invalid
```

```
        Else
                Return True ‘valid

End Function
```

This function ensures that no blank fields have inputas data.

Length Check

```
Private Function LengthCheck(variable, maxLength)

        If variable.Length > maxLength ‘if variable too long
                Return False ‘invalid
        Else
                Return True ‘valid

End Function
```

This function ensures that no fields are longer than their fixed length limits, to ensure that no data is stored in the field if it is too long, as this will result in the data being truncated to fit the limit.

Exact Length Check

```
Private Function LengthCheck(variable, length)

        If variable.Length = length ‘if variable is correct length
                Return True ‘valid
        Else
                Return False ‘invalid

End Function
```

This function checks that the variable is an exact length. This is used for values such as phone numbers and postcodes, which have to be a specific length.

Range Check

```
Private Function RangeCheck(variable As Real, min, max)

        If (variable < max) AND (variable > min) ‘if variable within bounds
                Return True ‘valid
        Else
                Return False ‘invalid

End Function
```

This function checks if a given number is within a certain range. If the number is too large or small, this function will return false.

Data Type Check

```
Function datatypecheck(variable As String)

        Try 'attempt to convert variable to integer
                Dim variableInt As Integer = variable
        Catch ex As Exception 'if cannot perform this, due to it not being a number
                Return False ‘invalid
         End Try

         Return True ‘valid

End Function
```

This function checks that a data input is a number, and does not contain non-numeric characters. This is to prevent letters being stored as numbers and then trying to use them in calculations, and this will break the system, as you cannot divide letters, for example.

Postcode Format Check

```
Private Function PostcodeCheck(variable)

        Declare valid As Bool = True
        Declare character As Char
        If variable.Length = 7   ‘if postcode correct length
                character = variable(x)
                For x = 0 To 6
                        If x < 3 OR > 5
                                If ASCII(character) > 63 OR < 54 ‘if characters 1,2,6,7 letters
                                        valid = False ‘invalid
                                End If
                        Else
                                If ASCII(character) < 64 OR > 89 ‘if characters 3,4,5numbers
                                        valid = False ‘invalid
                                End If
                        End If
                Loop
        End If

        Return valid
End Function
```

This function checks that a variable is exactly 7 characters long, and follows the AA000AA format, meaning it is therefore a valid UK postcode.

Phone Number Format Check

```
Private Function PostcodeCheck(variable)
```

```
        Declare valid As Bool = False
        If variable.Length = 11 'if number correct length

                Valid = DataTypeCheck(variable)  'if number is number

        End If

        Return valid
End Function
```

This function ensures that the data is exactly 11 digits, and only composed of numerical values, meaning it is a valid phone number. An additional check could be used to ensure the first two digits are 07, or 01978…, however there are too many regional and international codes to account for them all.

Double Check

```
Private Function DoubleCheck(variable, variableAttemptTwo)

        If variable = variableAttemptTwo
                Return True
        Else
                Return False

End Function
```

This function verifies that the user has correctly input the data a second time, to ensure that any double check verifications match. This is used when creating passwords, for example, to reduce the chance of entering the password incorrectly due to a typing error,

Rule Applied	**Fields Used In**	**Why?**	**Error Message**
Must be unique	New Username	As the username is the primary key of the users.txt file, it must be unique.	"That username is already in use. Please try a different one."
Must be less than X characters long	All stored data inputs *New Customer Name* *New Customer Address* *New Password* etc.	As the data is stored in fixed length files, the contents of the data will be lost if it exceeds the character limit of the field, and so the data must be less than the limit.	"Data cannot exceed X characters."
Numerical	Job Length	As this value is used in calculations (such as the timetable generation	"Value must be numerical."

		calculation) then it must be a valid number, and if it is not, it will cause a runtime error.	
Must be 00AAA00 format	New Customer Postcode	A postcode must be of the 00AA00 format (e.g LL115FD) to be valid.	“Invalid postcode. Please try entering it again.”

Data Structures for non-volatile storage

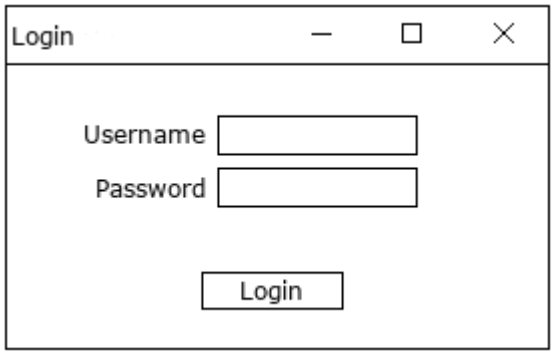
```
Public Structure User
    Public Username As String       'length 12; start 1 ; end 12
    Public Password As String       'length 12; start 13; end 24
    Public AccessLevel As String    'length 1; start 25; end 25;
  End Structure

Public Structure Customer
    Public ID As String             'length 3 ; start 1 ; end 3
    Public Name As String           'length 25; start 4 ; end 28
    Public Number As String         'length 11; start 29; end 39
    Public Address As String        'length 25; start 40; end 64
    Public Area As String           'length 12; start 65; end 76
    Public Postcode As String       'length 7 ; start 77; end 83
  End Structure

Public Structure Job
    Public ID As String             'length 3 ; start 1 ; end 3
    Public name As String           ‘length 20; start 4 ; end 23
    Public rppu As Real             ‘length 6 ; start 24; end 29
    Public lppu AS Real             ‘length 6 ; start 30; end 35
    Public resources As String      ‘length 40; start 36; end 75
  End Structure

Public Structure Materials
    Public ID As String             'length 3 ; start 1 ; end 3
    Public name As String           'length 25 ; start 4 ; end 28
    Public ppu As String            'length 6 ; start 29 ; end 34
  End Structure

Public Structure Booking
    Public ID As String             'length 3 ; start 1 ; end 3
    Public customerID As String     'length 3 ; start 4 ; end 6
```

```
    Public jobID As String          'length 3 ; start 7 ; end 9
    Public jobLength As String      'length 3 ; start 10; end 12

    Public area As String           'length 12; start 65; end 76   (FROM CUSTOMER FILE)
    Public postcode As String       'length 7 ; start 77; end 83   (FROM CUSTOMER FILE)
End Structure

Public Structure TimetableItem
    Public ID As String             'length 3 ; start 1 ; end 3
    Public bookingID As String      'length 3 ; start 4 ; end 6
    Public day As String            'length 3 ; start 8 ; end 10
    Public startTime As String      'length 3 ; start 12; end 14
    Public jobLength As String      'length 3 ; start 16; end 18
End Structure

Public Structure TimetableItem
    Public ID As String             'length 3 ; start 1 ; end 3
    Public bookingID As String      'length 3 ; start 4 ; end 6
    Public day As String            'length 3 ; start 8 ; end 10
    Public startTime As String      'length 3 ; start 12; end 14
    Public jobLength As String      'length 3 ; start 16; end 18
    Public week As String           ‘length 10; start 21;
End Structure
```

Detailed Design

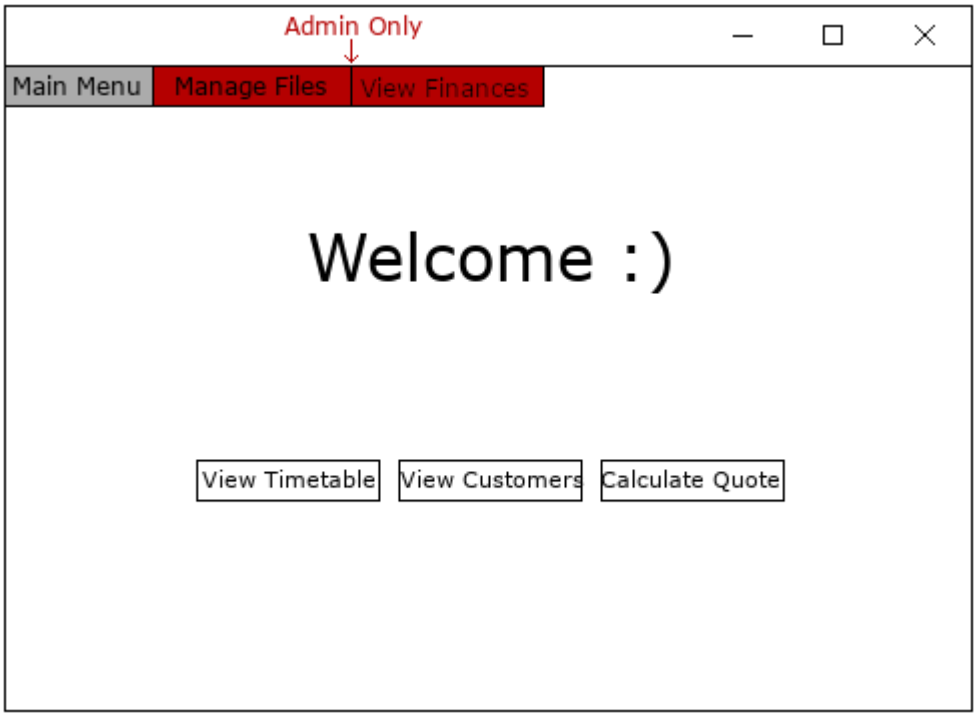
Inputs

Input logins details to access program.



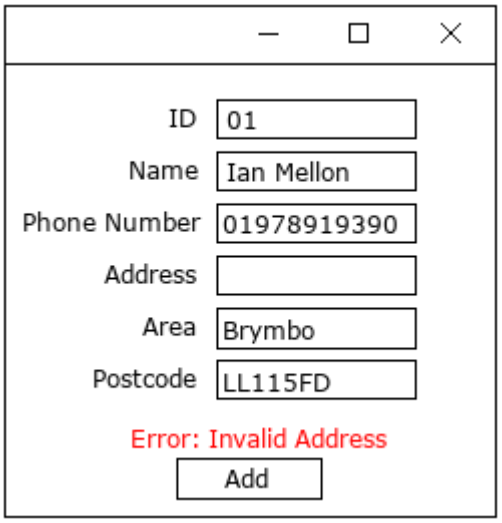
Text boxes will be used for the username and password input fields, marked by labels corresponding to them. A button will be used to trigger the data comparison (between the inputs and stored data in the file) and then either proceed to the menu, or relay an error message to the user. Although not evident on this diagram, error messages will be displayed via a red label just above the button (please refer to the input form below for a visual representation).

Main menu screen to navigate the system.



At the top are three buttons, however, these will only be present during an admin session. If a regular user is logged on they will only see the bottom buttons. These top buttons will be used to navigate to admin-only sections of the form. The bottom buttons will also redirect the user to other functions. In the middle is a label that greets the user. Whilst this is mostly there as a pleasantry, it can be used to hold other information, for example, if the system is updated one day, important information about the system can be displayed there for the user.

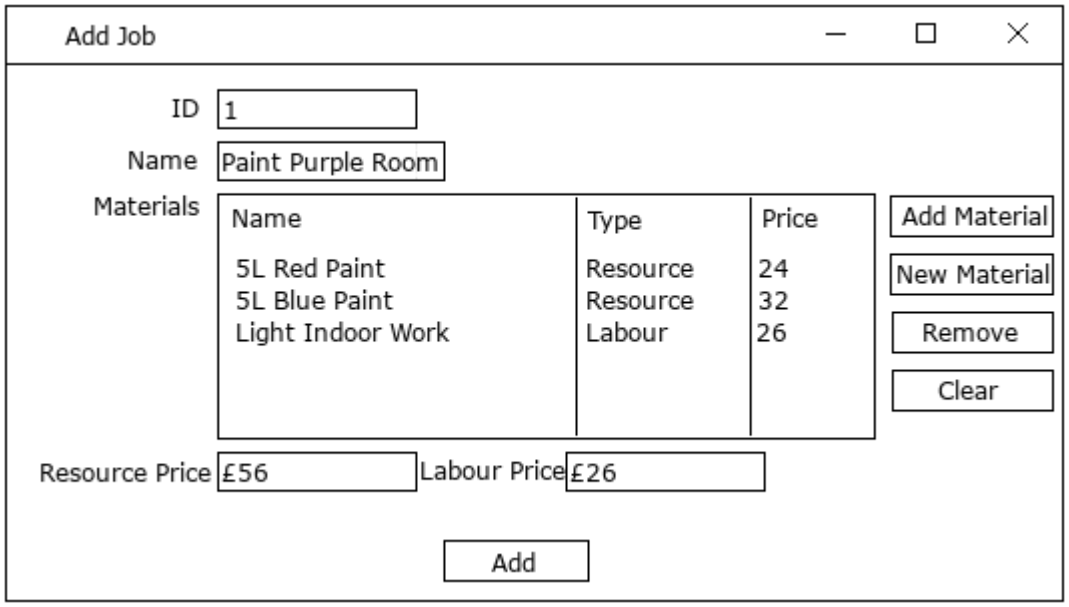
Inputting data to be stored in a file. All of the forms to do this will be very similar, with the only difference being 1) the number of fields, and 2) the names of said fields. This example form is for adding a customer.



This form makes use of text boxes to input into, labels to mark the boxes' purposes, a button to trigger the validation/storing processes, and an error label (the same as

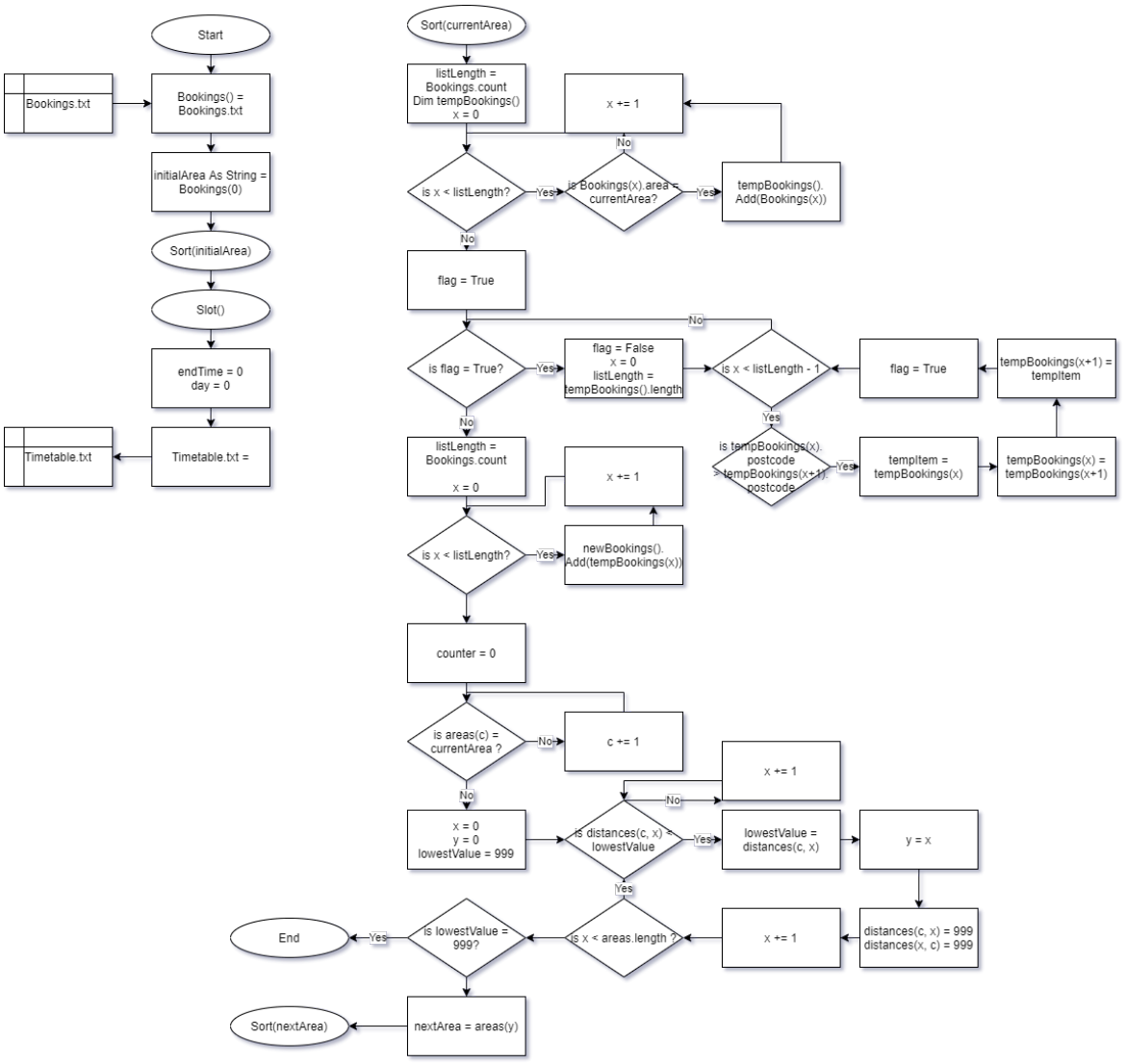
the one mentioned above). All of the data input forms will be like this, except with varying numbers of boxes, different labels, and, in some cases, will make use of other objects such as comboboxes. This specific example is for adding customers.

Other data entry forms will be very similar to this. The forms for adding users, bookings and materials will be the same as this form, except the number of boxes and names of the labels. Add Job however will be different, as shown below:



This form will require users to add items to a list using button controls. Pressing ‘Add Material’ will open a list of all the materials inmaterials.txt, and require the user the select one, which will add it to the list shown above. The prices will be automatically calculated.

Timetable Generation
Sorting the Bookings into Order



This flowchart shows code to sort a list of data (retrieved from bookings.txt) twofold, so that 1) all areas are grouped together, and 2) all items within those groups are sorted by postcode. The diagram on the left is the main procedure. The diagram on the right is the sorting algorithm (making use of a bubble sort), which is recursively called until the entire list has been sorted. The new sorted list is stored in a new list called tempBookings. Below is an example of this process.

Unsorted

Below is some example data composed of the area, postcode and length of the booking. (This data will be stored in the customers and jobs file, and accessed using foreign keys, but for this example that is being ignored).

ID	Area	Postcode	Length
0	Brymbo	LL115FD	1h
1	Marford	LL128TX	2h
2	Borras	LL139QW	45m
3	Brymbo	LL116AB	30m
4	Borras	LL139QW	1h
5	Wrexham	LL12BG	1h
6	Marford	LL128SZ	30m
7	Brymbo	LL115FD	15m
8	Borras	LL112AB	2h

Grouped

The data is then sorted into groups by area, putting all the Borras bookings together, Brymbo together, etc. This is done by using a simple bubblesort.

ID	Area	Postcode	Length
2	Borras	LL139QW	45m
4	Borras	LL139QW	1h
8	Borras	LL112AB	2h

0	Brymbo	LL115FD	1h
3	Brymbo	LL116AB	30m
7	Brymbo	LL115FD	15m

1	Marford	LL128TX	2h
6	Marford	LL128SZ	30m

5	Wrexham	LL12BG	1h

Grouped (Sorted)
Each group of data is then sorted by postcode, grouping the same postcodes together. The list is now sorted two-fold.

ID	Area	Postcode	Length
8	Borras	LL112AB	2h
2	Borras	LL139QW	45m
4	Borras	LL139QW	1h

0	Brymbo	LL115FD	1h
7	Brymbo	LL115FD	15m
3	Brymbo	LL116AB	30m

6	Marford	LL128SZ	30m
1	Marford	LL128TX	2h

5	Wrexham	LL12BG	1h

Sorted
The lists are then merged into one list, in an order determined by the least distance algorithm. The table below is used to determine the order.

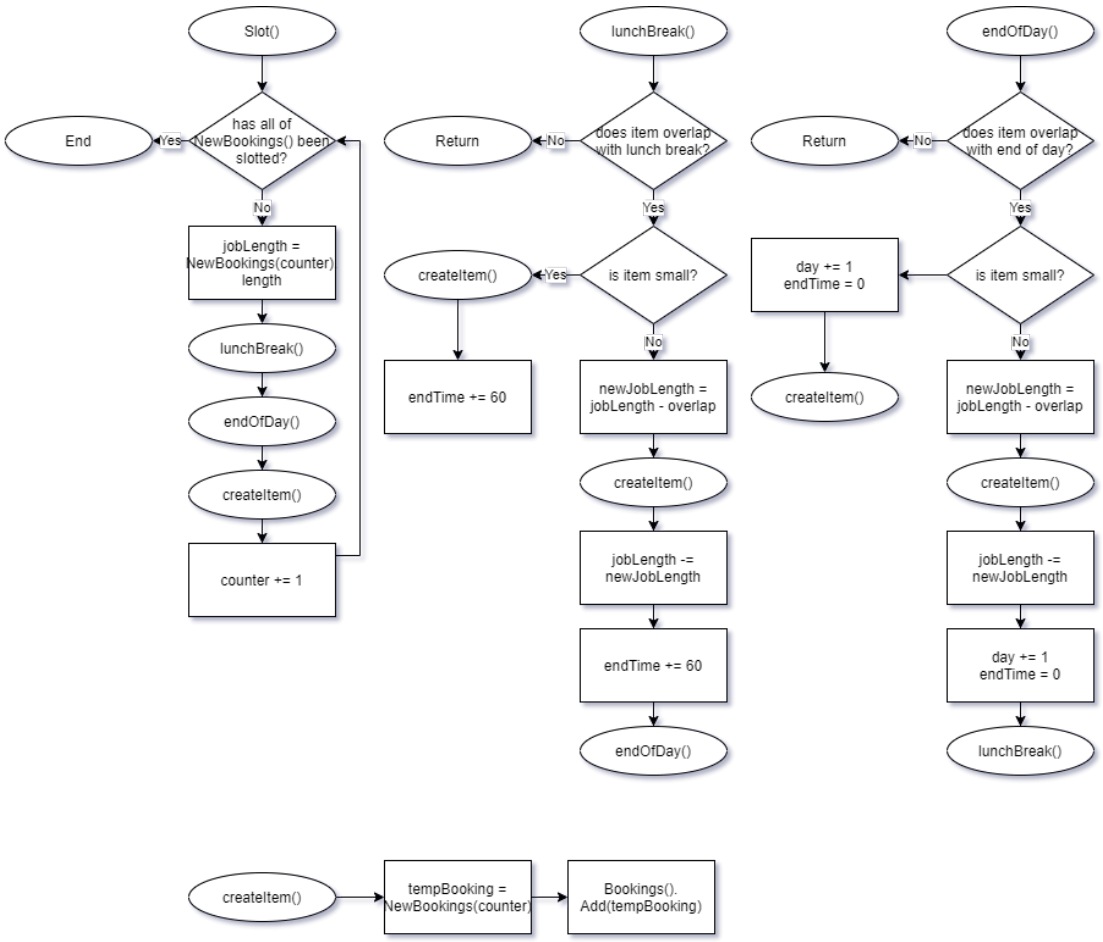
(**Distances**. This table has the drive-time between each area. The algorithm will start with the first group (Borras in this case). The algorithm will then search for the closest place to Borras: Wrexham. Then the closest to Wrexham (that hasn't already been used) is Brymbo. etc.)

	Borras	Brymbo	Marford	Wrexham
Borras	/	13	10	7
Brymbo	13	/	15	9
Marford	10	15	/	12
Wrexham	7	9	12	/

This sorted will result in the final order below.

ID	Area	Postcode	Length
8	Borras	LL112AB	2h
2	Borras	LL139QW	45m
4	Borras	LL139QW	1h
5	Wrexham	LL12BG	1h
0	Brymbo	LL115FD	1h
7	Brymbo	LL115FD	15m
3	Brymbo	LL116AB	30m
6	Marford	LL128SZ	30m
1	Marford	LL128TX	2h

Slotting the Items into a Timetable



The diagram above is what the class uses to then fit the sorted data into days of the week. This prevents any items from going over the end-of-day limit (for example, 18:00) by either ending the day slightly early, and moving the overlapped item to first-thing next-day if it is small, or, if the overlap is too large, the item remains on the original day, but is cut short, and then is continued later next-day. This system also forces a gap to appear (for example for 1 hour, around noon) for lunch, using similar procedures. The ‘split’ list is then stored into the NewBookings list, which is then saved into timetable.txt.

Below is an example of the above table (from the sorting function) being slotted into a timetable.

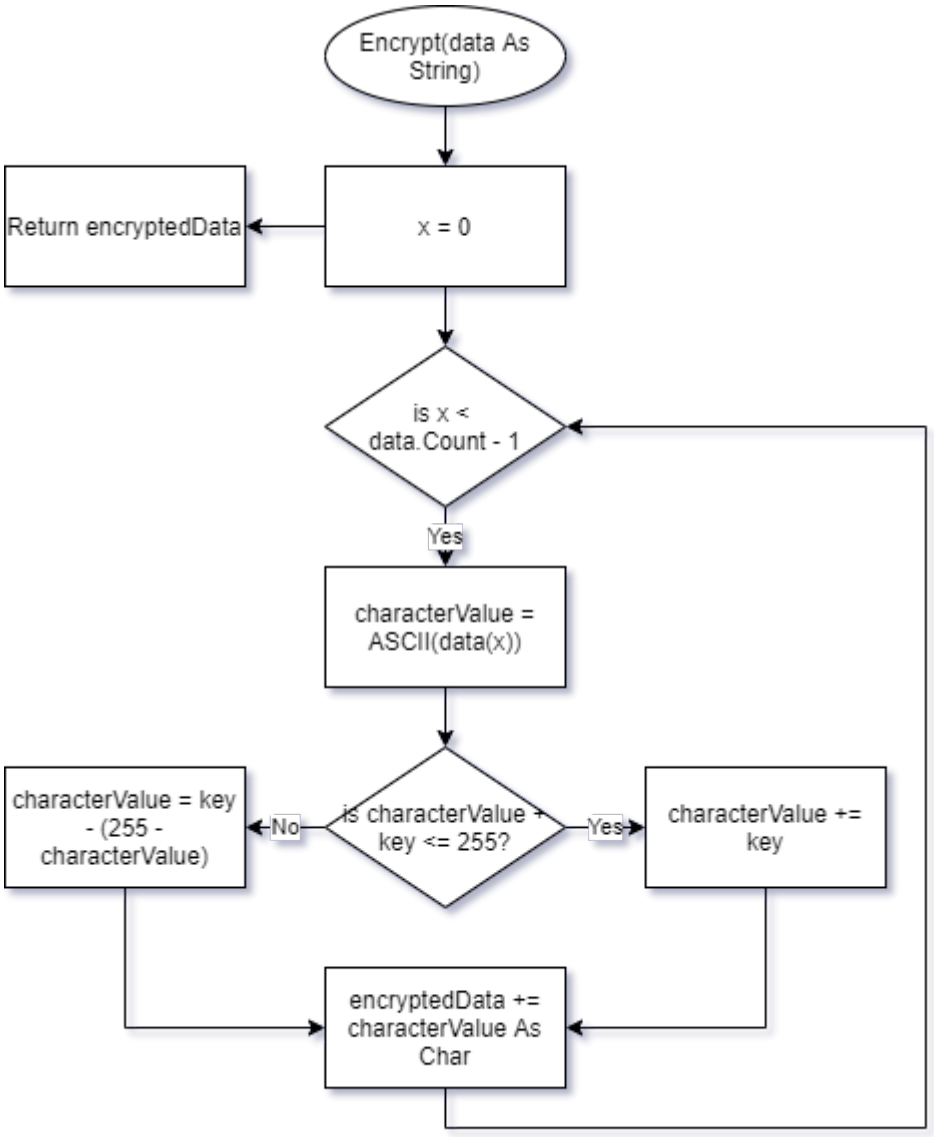
	Monday	**Tuesday**
8am		
		0

	8 (Borras)	(Brymbo)
9am		7 (Brymbo)
		3 (Brymbo)
		Brymbo → Marford
10am	2 (Borras)	6 (Marford)
		1 … (Marford)
	4 (Borras)	
11am		
	Borras → Wrexham	
12	Lunch	Lunch
1pm	5 (Wrexham)	… 1 (continued)
2pm		

Item 1 is an example of the lunch splitting, as the job has been split to go before and after lunch, and not overlap with it.

Encryption

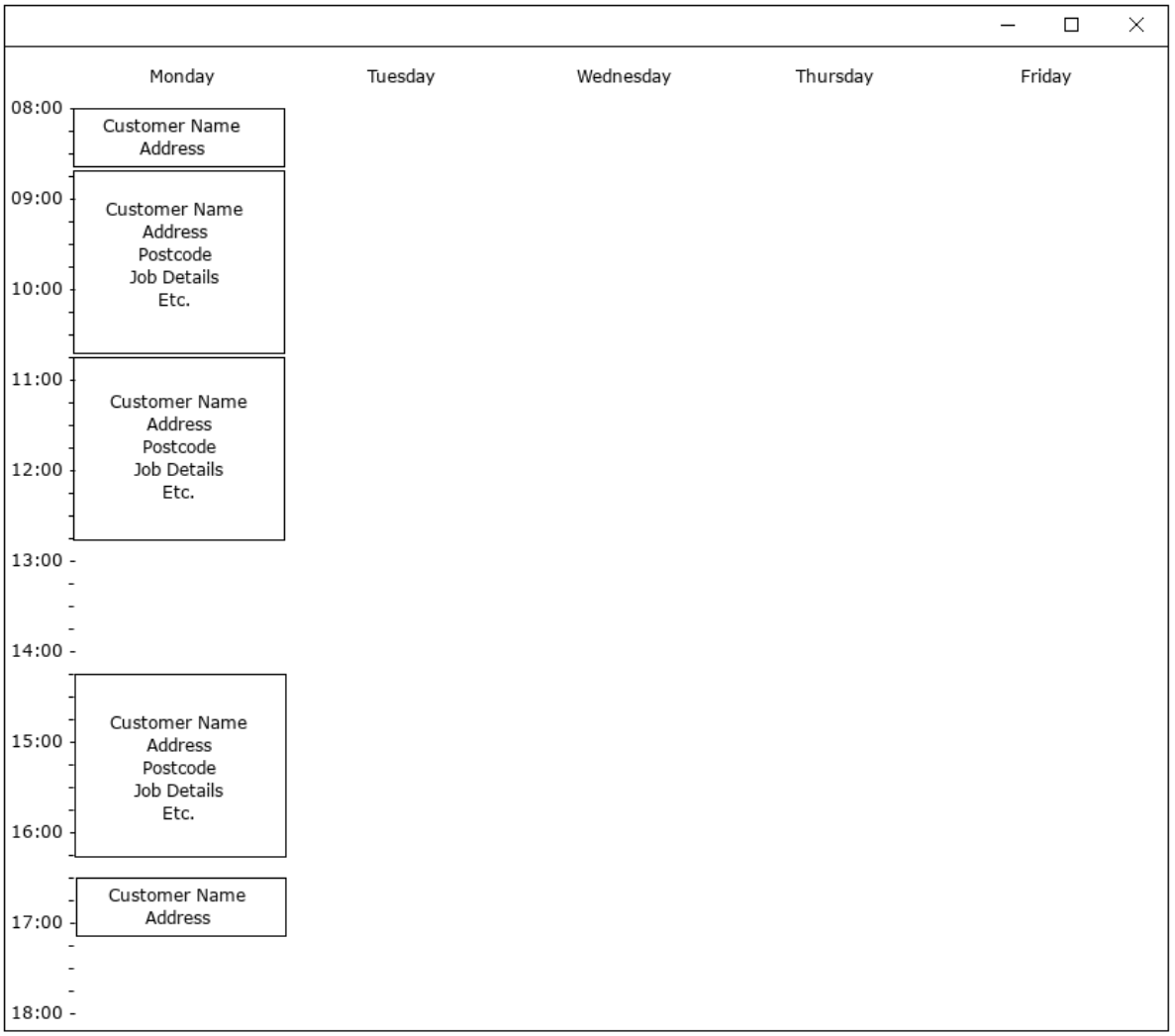
Encryption (via Caesar Cipher), where ‘key’ is a set number of characters which the letters need to be shifted by.



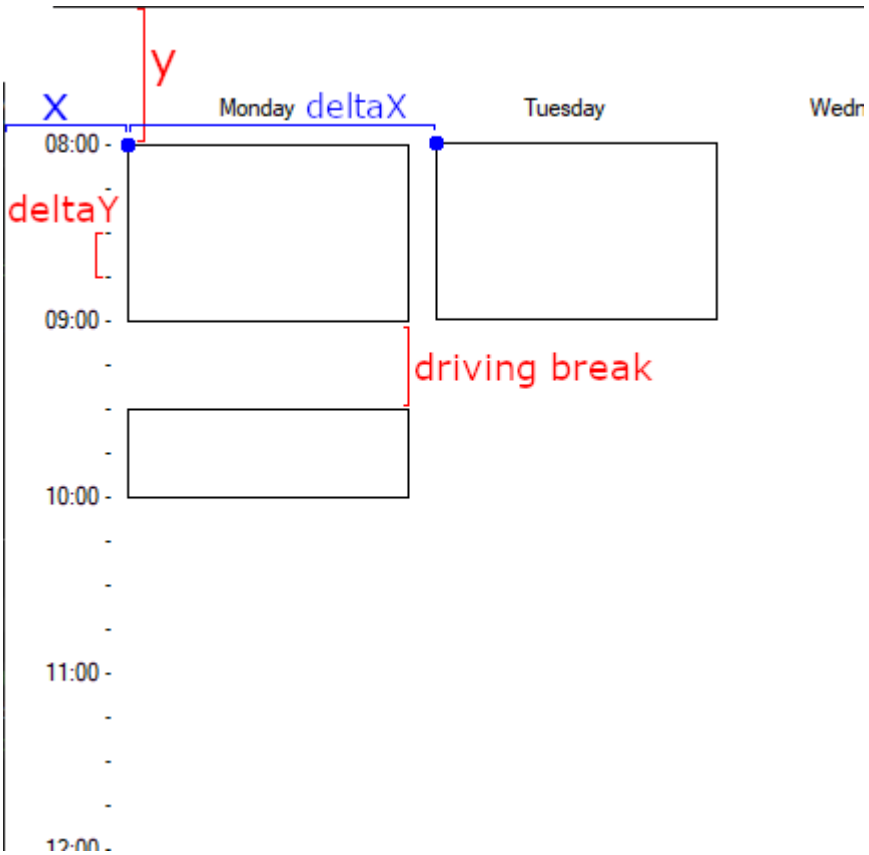
The data (such as the username, or password) is inputted into the function as a parameter. The system will then run a loop for each character in that variable. The function will convert the current character *(A)* into its ASCII numerical value *(64)*, and then add the key value to it (such as 42, for example) *(106)*. The system then converts the new character code back into an actual character *(j)*, and then adds it back into a new variable. Once this loop is completed, the new variable is the full word, but shifted, and the function can return that value.

Outputs

View Bookings via a timetable



This form outputs the timetable in a visual style for the user to see, using buttons and labels. On the x and y axises, labels are used to mark out the days and times respectively. Buttons will be instantiated, sized and positioned to show their time and length of occurrence. The button text is used to display the information of the booking. However, for smaller bookings (such as the bottom booking) not all of the information fits. When a button is pressed, it will show the user all of the booking's information, so that any data that does not fit is still accessible to the user.



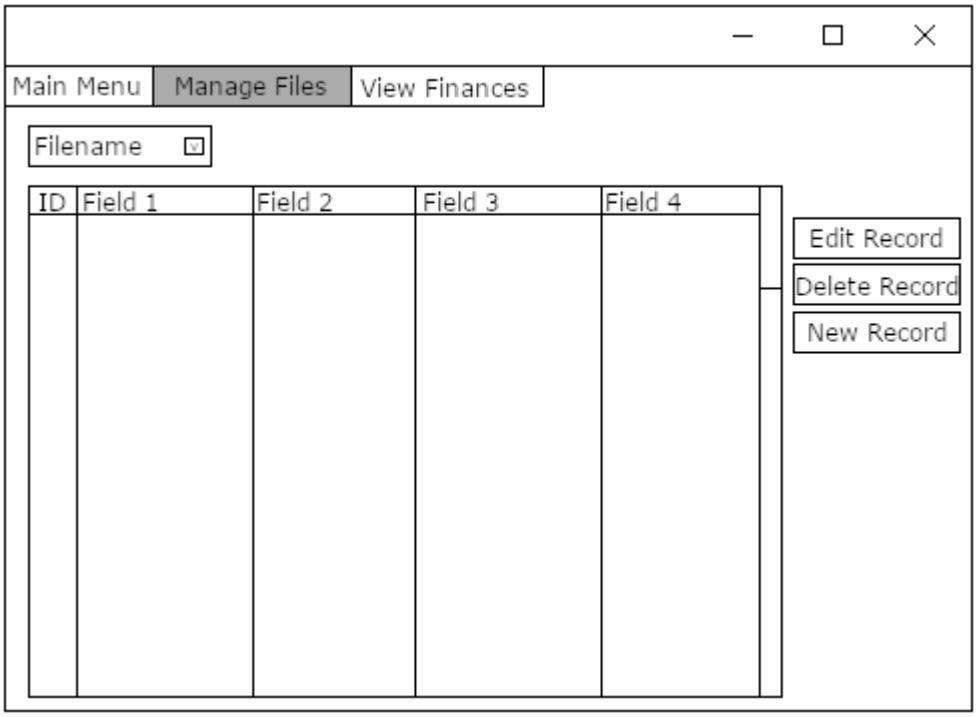
The coordinate point of a button is the top left corner (marked in blue).The top-left-most blue point will be the origin, at coordinates (x, y). deltaX and deltaY will be set values that will be decided once production has begun.

To determine the location of a button the following algorithm will be used:

```
Dim coordinate As Point

coordinate.X = origin.X + deltaX(day) 'Monday = 0, Friday = 4
coordinate.Y = origin.Y + (deltaY(startTime / 15)) '8am = 0, 9am = 60, etc

button.Position = coordinate
button.Length = (length / 15) * deltaY
```

button.Width will be a fixed value, which will be determined during production.

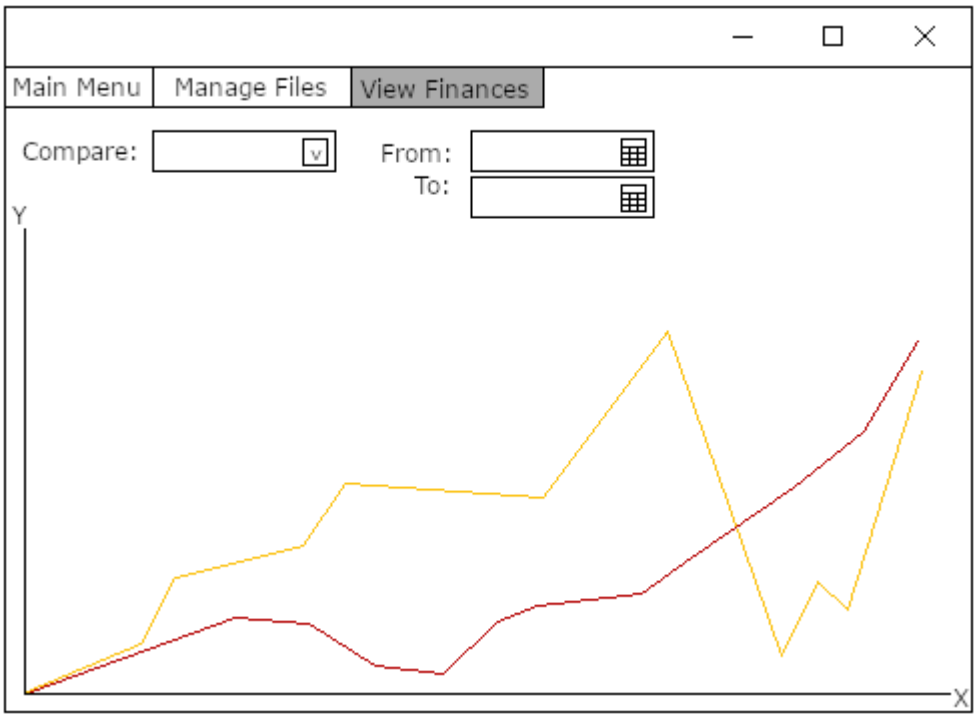
View, and edit, file contents. (Admin only).



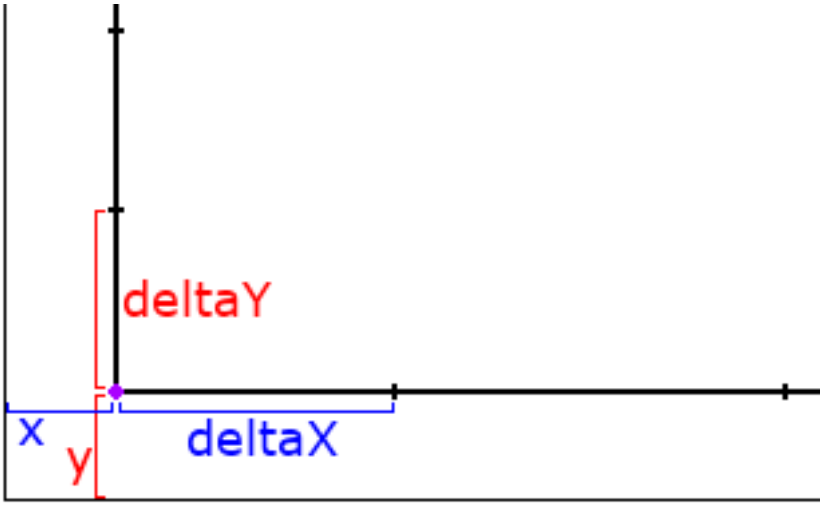
This form is a variation of the menu screen, and is only accessible via the admin-only top buttons. Below them is a combobox called ‘Filename’. This box will display all of the files in use by the system, and will allow the user to select which file they would like to manage. Once selected, the list view below will show the contents of the selected file. The listview will be intractable, and so data can be selected by clicking on it. Although not visible here, a search bar will appear (which will consist of a label, textbox, and button). This will be used to search a file and only display matching results. This form also has buttons on the right hand side. These are used to manipulate the data - for example editing, or deleting a record - or, adding a new record to the system.

The edit record will open the ‘Add’ forms from above, but will fill the fields with the relevant data, and instead of amending the record to the file, it will overwrite the existing file with the new information.

View a comparison of different things, such as the money earned, between months. (Admin only)



This form will once again be only accessible to admins. This form consists of labels, a combobox, and date boxes at the top. This will allow the user to select what they wish to compare, and when they would wish to compare it from. Below is the graph, which will be made of labels and graph tools. The graph will automatically generate once it detects all necessary fields have been filed, and will update when one is then changed, meaning no button is required to prompt generation.



The pink point will be the origin, at coordinates (x, form.Height - y). deltaX and deltaY will be fixed values decided during production. There will be 5 notches along each

axis. The maximum value of Y will be calculated before the graph is drawn, using a simple loop and if statement.

```
Dim maxValue = items(0)

For x = 1 to items.Count
        If items(x).price > maxValue
                maxValue = items(x).price
        End If
Next
```

To determine the location of a point on the graph the following algorithm will be used:

```
Dim coordinate As Point

coordinate.X = origin.X + deltaX(numberOfDays / 5)
coordinate.Y = origin.Y - deltaY(maxValue / 5)
```

Then the graph can either create a bar chart or line chart depending on which is needed, using the graphics and pen functions. For a bar chart:

```
Dim surface As Graphics = CreateGraphics()
Dim pen As Pen = New Pen(Color, 2)

surface.DrawLine(pen, coordinate.X, origin.Y, coordinate.X, coordinate.Y)
```

This will draw a straight vertical line from the x-axis to the point.

For a line graph:

```
Dim surface As Graphics = CreateGraphics()
Dim pen As Pen = New Pen(Color, 2)
Dim oldCoordinate As Point = origin

For items
        surface.DrawLine(pen, oldCoordinate.X, oldCoordinate.Y, coordinate.X,
coordinate.Y)
        oldCoordinate = coordinate
Next
```

This will draw a line from the origin to the first point, and then a second line from the first point to the second point, etc. creating a graph such as the one above.

Software Development

Copy of **README.txt**

DEFAULT LOGIN DETAILS
USERNAME: admin
PASSWORD: root

a list of other users is visible by selecting 'Manage Files' and then selecting 'Users.txt'.

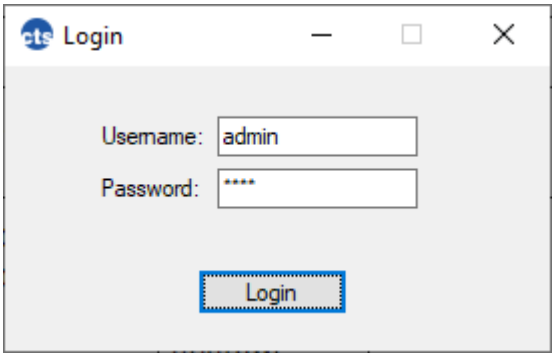
IMPORTANT!!!!
The view timetable feature is date-sensitive, how this works is:
- any past weeks will display only completed jobs
- future weeks will display only the current timetable
- present weeks will show completed jobs for days that have passed this week, and the current timetable for the present and future day. FOR EXAMPLE, if looking on a Wednesday, Monday and Tuesday's items will only show confirmed bookings. To fully test this, you will need to either look at multiple days, or, change your system's date to experience the difference of different days.

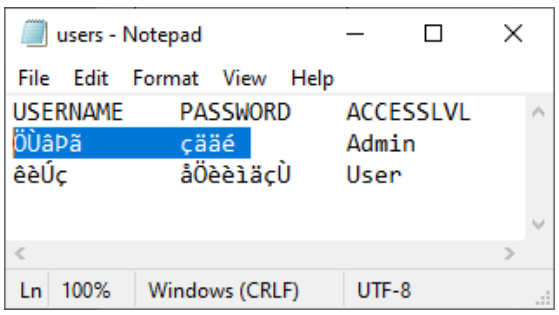
(PAST ITEMS ARE A LIGHTER GREY TO HIGHLIGHT THIS)

The 'Job Completed' and 'Day Completed' buttons will save the relevant items as completed, and so will still be visible after the day has ended. Any items not marked will no longer display once the day has ended. NOTE: only the jobs of the current day can be marked as completed.

All files can be found in the default location.

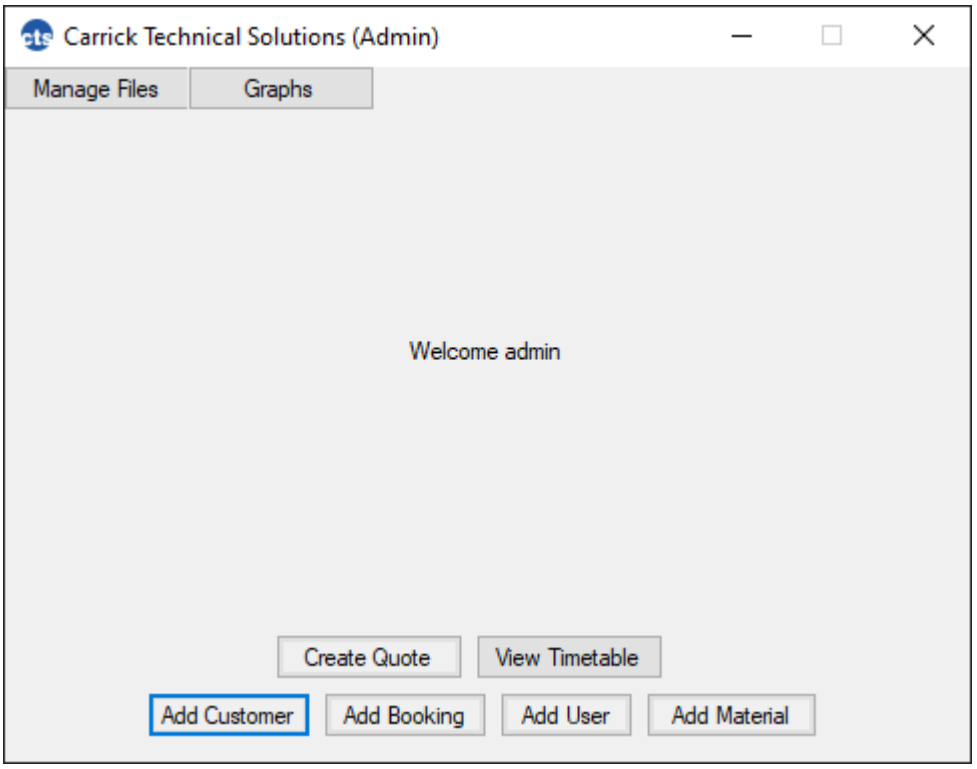
Login (Gate.vb)
This formed is used to login to the system, only gaining access to the other forms if the username and passwords are valid.



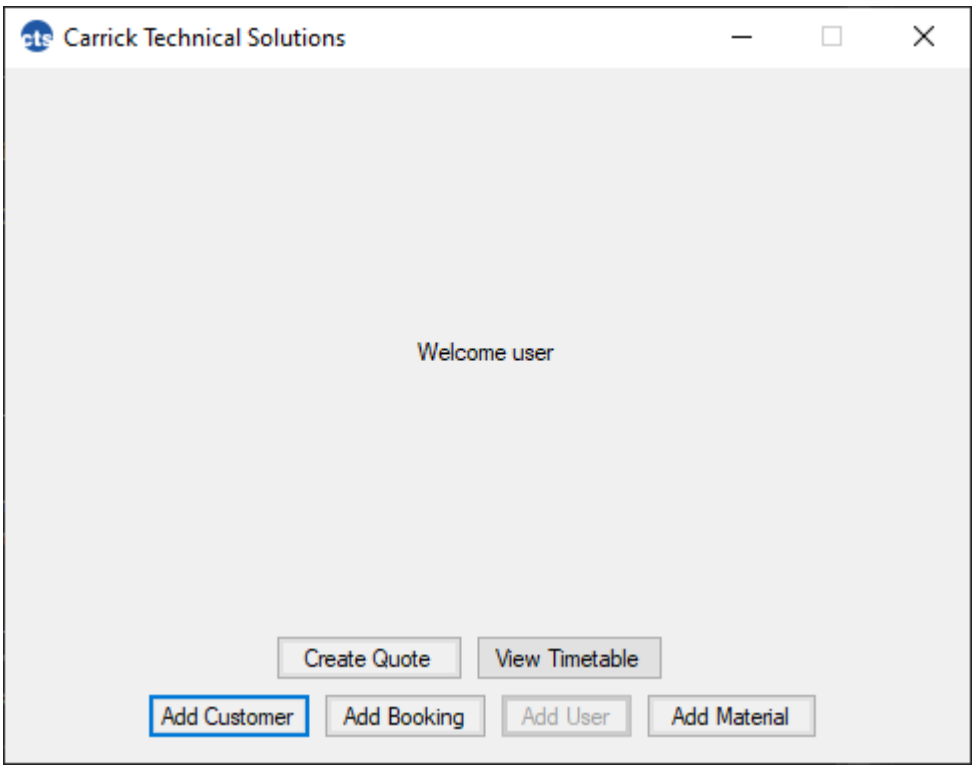


Main.vb

This form is the central navigation form, which links to the other forms of the system, allowing the user to access them by clicking the respective button.

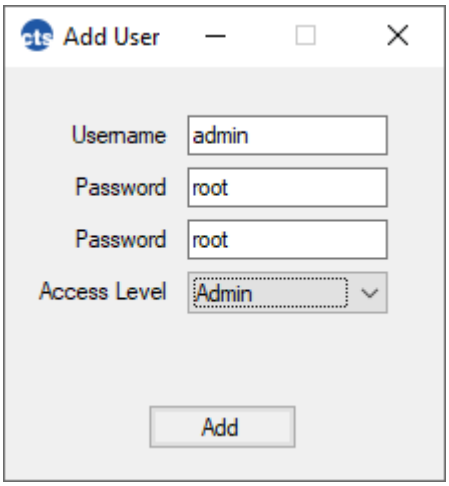


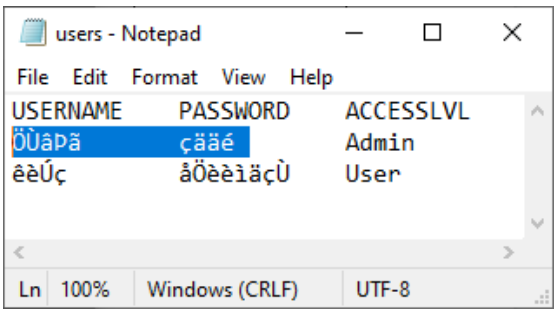
The above screenshot is when an admin logs in. The below images when a standard user logs in. This shows how the access-level system works.



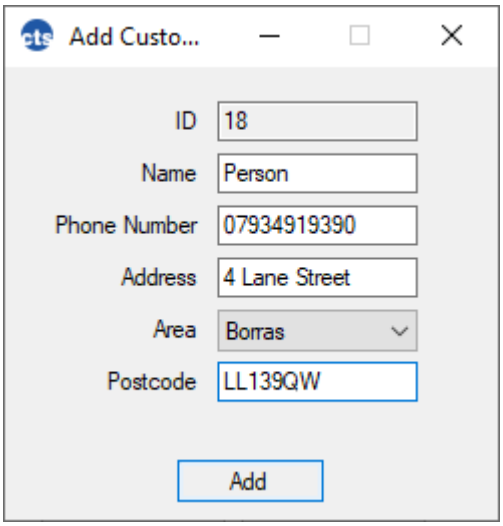
AddUser.vb

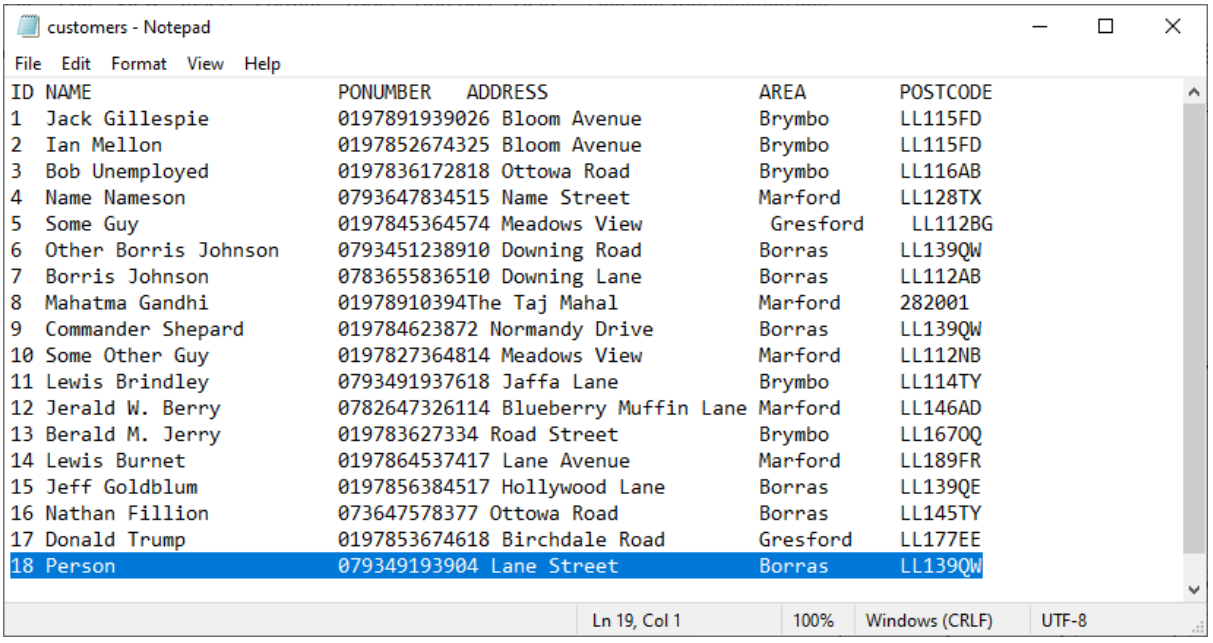
This form allows admins to input the login details for new users, which will be able to access the system using the details the admin inputs.





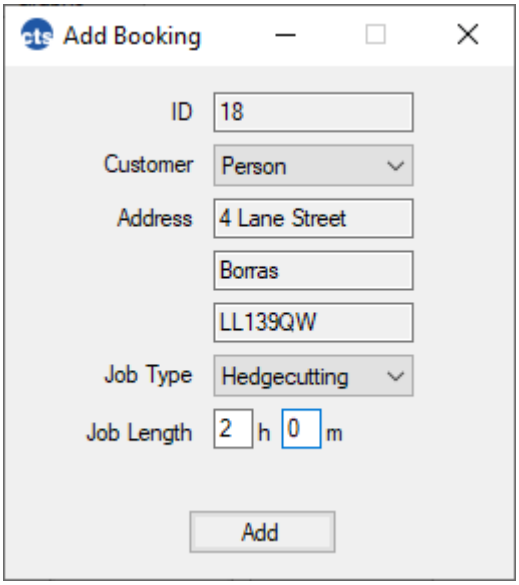
AddCustomer.vb

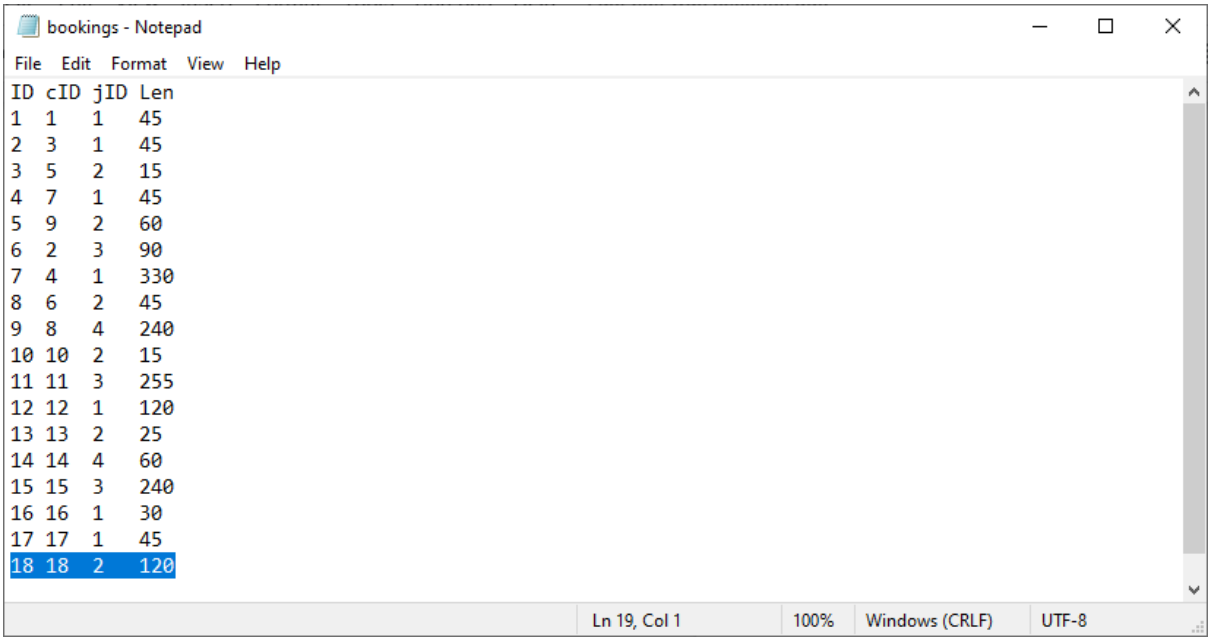




This form allows users to input the details for new customers, which will be stored in the system, allowing users to view the details, and the system to use their data in calculations, such as when determining the route for the timetable, using their addresses.

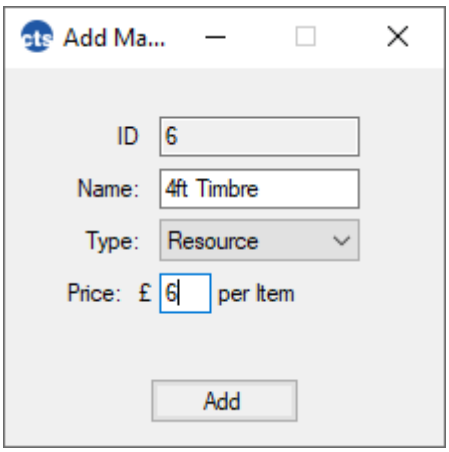
AddBooking.vb

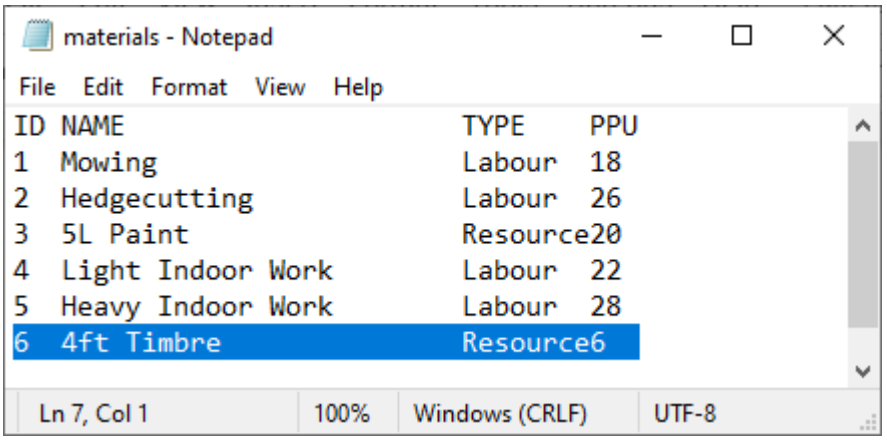




This form allows users to input the details for new bookings, which will be stored in the system.

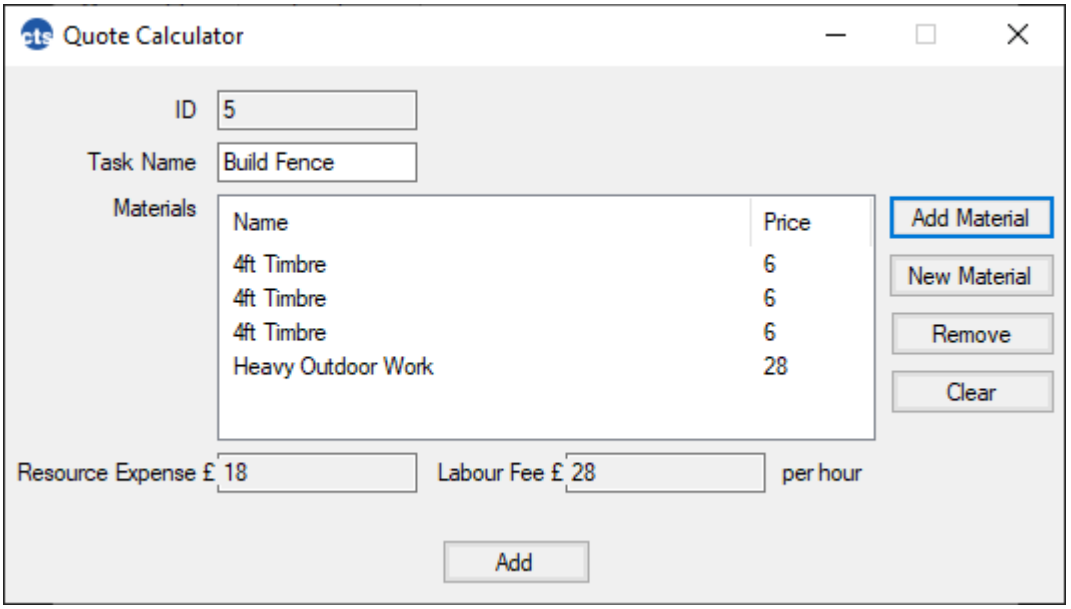
AddMaterial.vb

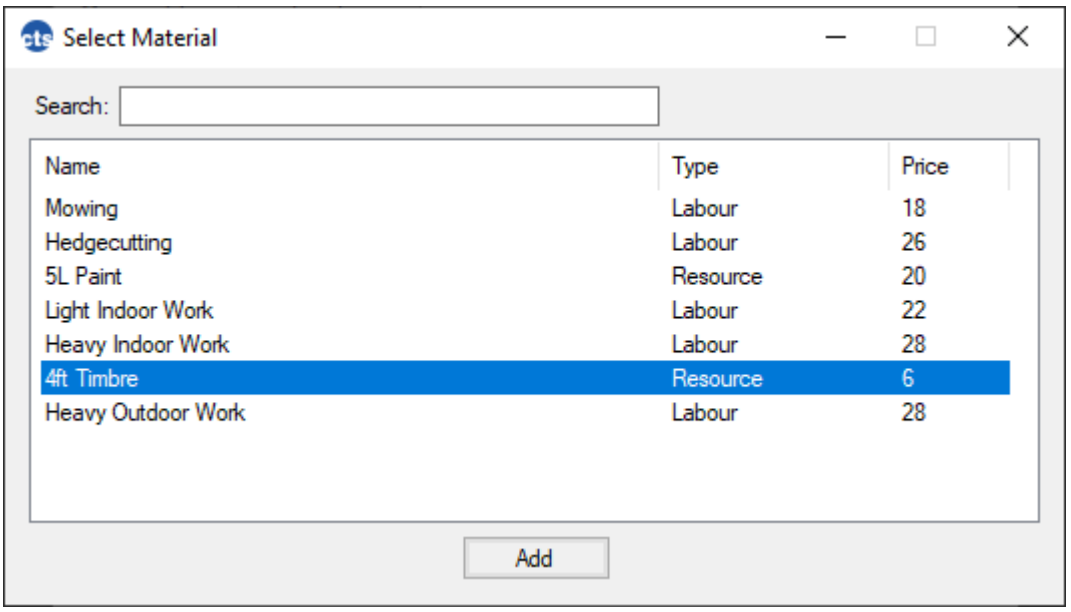


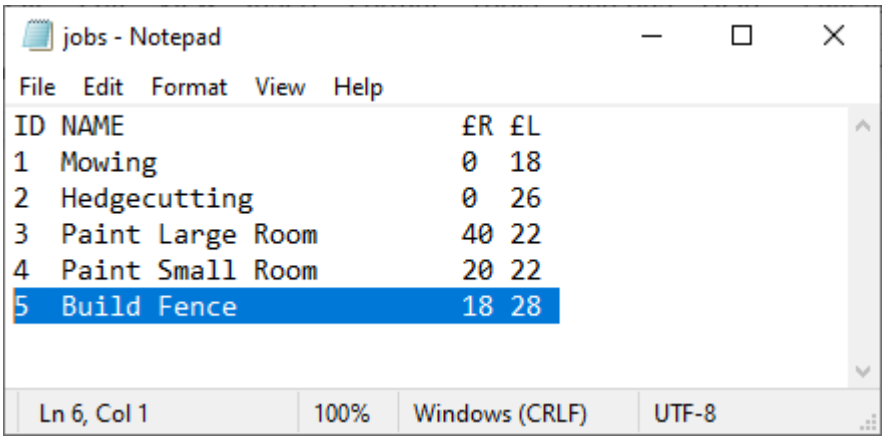


This form allows users to input the details for new customers, which will be stored in the system

AddJob.vb (and **SelectMaterial**.vb)

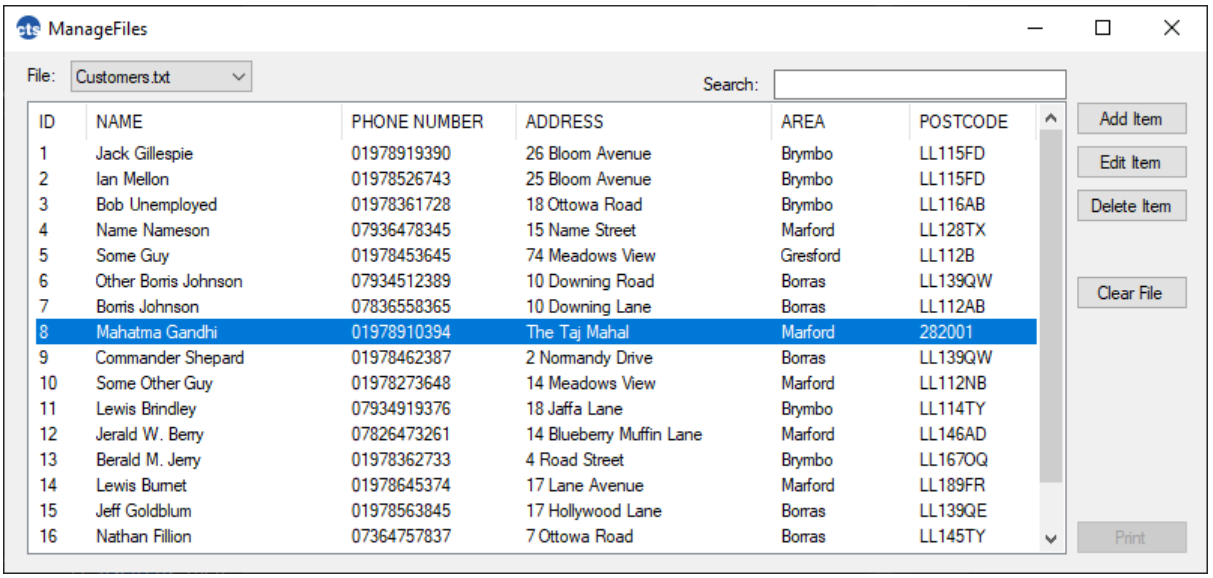






This form allows users to input the details for new jobs, which will be stored in the system, and used in other forms, such as AddBooking and in the monetary calculations of ViewTimetable and Graph.

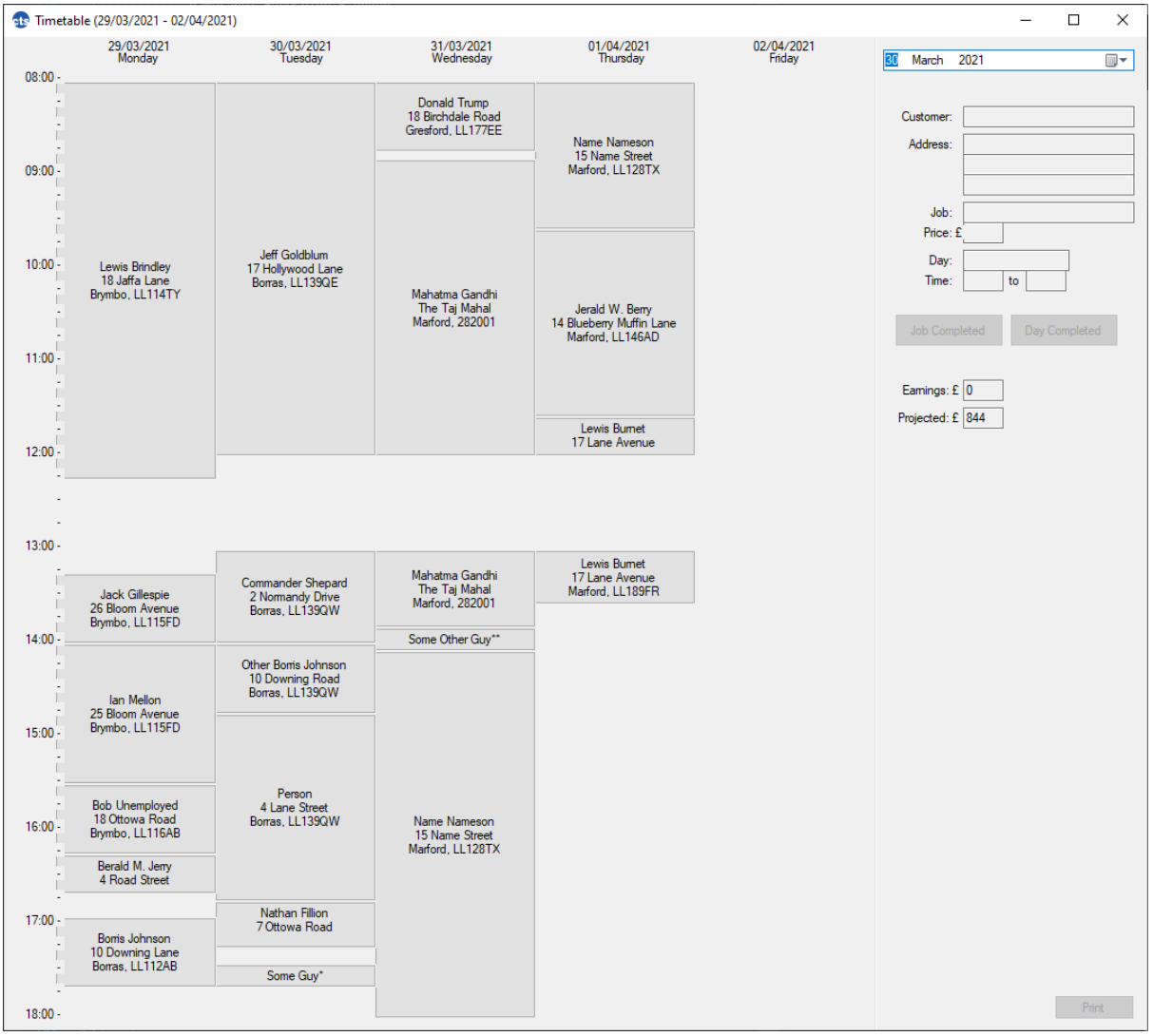
ManageFiles.vb



This form allows admins to view the contents of the files (with foreign keys being displayed as the data they reference,andthe login details being decrypted). This will allow admins to review data. As well as this, the admins will be able to add new items (using the Add forms), edit existing items (using an altered version of the Add forms), delta items from the files and can perform a total erasion of the files' contents. The controls for these are on the right.

This form also includes a search bar, which allows for users to easily locate specific items, by typing data the item contains, instead of having to manually scroll through all the records.

ViewTimetable.vb

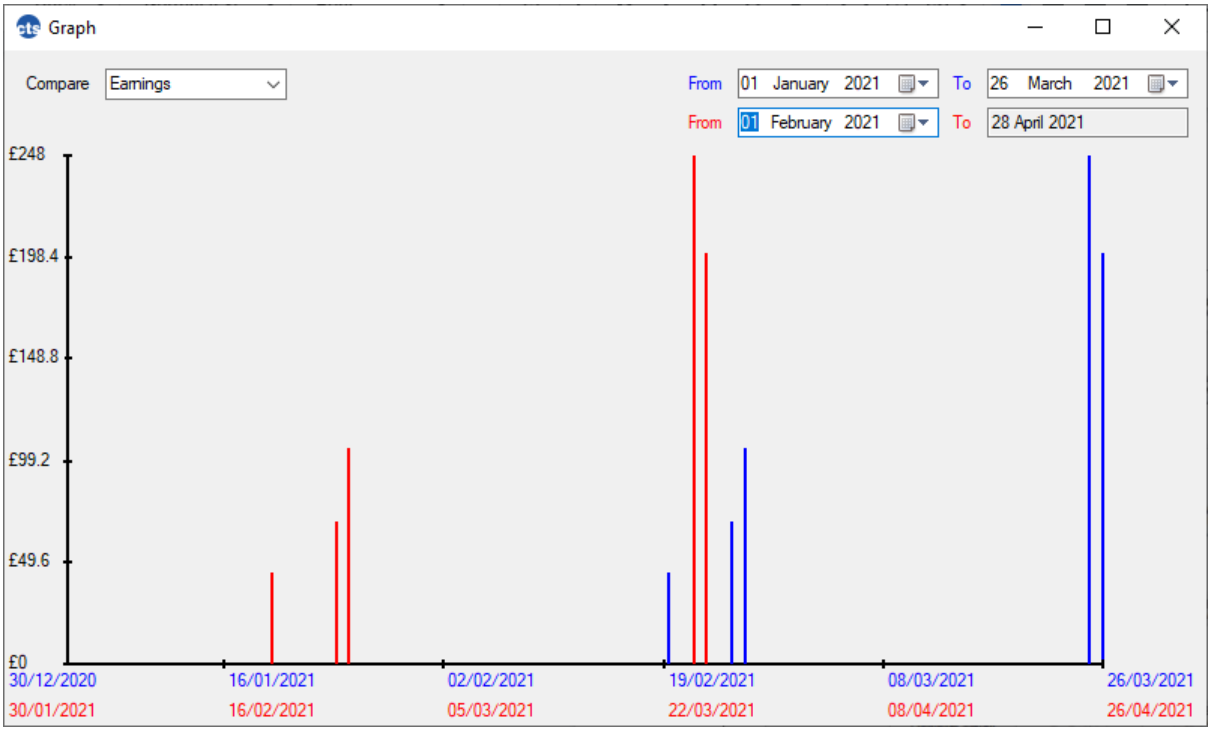


This form allows users to view the timetable generated by the system in the form of a visual table. The table consists of buttons which display basic

information on them (as shown above), but when selected, will display the specific data in the fields in the top right of the form, allowing the user to view less common data easily, and without it cluttering the form by being in the button's text.

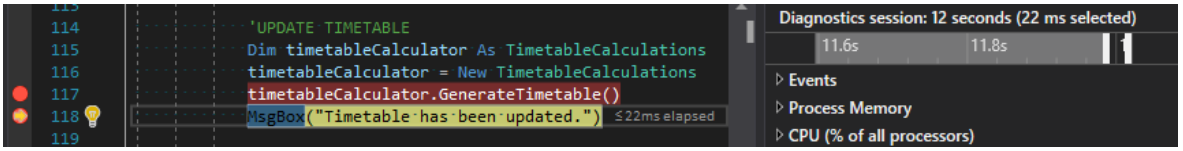
Also on the right, the form displays how much money the week should earn if all jobs are completed in the projected earnings field, and will display the sum of money that has actually been earned so far in the current week, in the earnings field.

Graph.vb



This form allows the user to view the earnings of days within two time periods, and can be used to compare the results, which are shown simultaneously.

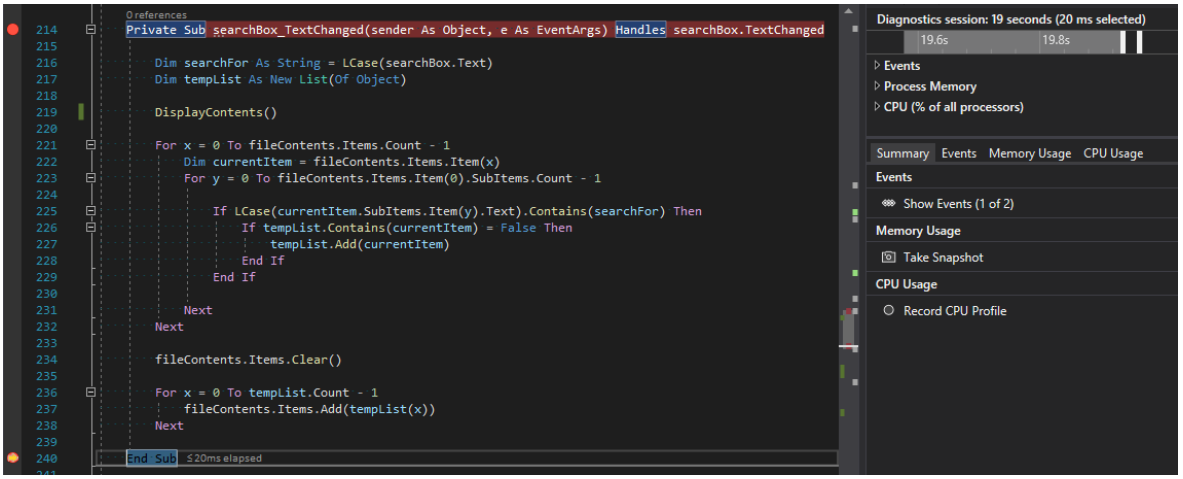
No.	Desc.	Done?	
1	It should be significantly faster to generate a timetable.	Yes	The timetable generation took around 22ms to complete when testing, which is significantly faster than Shaun's half-an-hour manual process.



Above is the code responsible for triggering the timetable generation. This screenshot shows that the time between these breakpoints (just before the process began, and after it was completed) took a time of 22ms on my computer. This time will vary depending on several factors, such as the specs of the computer, and the amount of bookings the algorithm has to handle, however, from my investigation I found that the old system could take Shaun over half an hour to perform, meaning even if his device cannot perform this task as fast as mine can, it is almost guaranteed that it will be faster than the old system. Another note is that as well as being faster, this solution is also personally easier for Shaun, as the computer handles all the calculations, and all he has to do is press a button.

The timetable generation is performed by the TimetableCalculations.vb class, which is called by the AddBooking form on line 117, and the ManageFiles form on line 290.

2	It should be faster to find a customer's details.	Yes	Using the search bar feature in the Manage Files form took around 20ms during testing, whereas searching through the paper records could take several minutes.



The above screenshot shows that it takes a total of 20ms to run the whole search function. According to my investigation research, it could take several minutes for the employees to search through the paper records and find the correct customer, which is a substantially longer amount of time.

```
216  Private Sub searchBox_TextChanged(sender As Object, e As EventArgs) Handles searchBox.TextChanged
217
218      Dim searchFor As String = LCase(searchBox.Text) 'text to search for (vonverted to upper case)
219      Dim tempList As New List(Of Object) 'stores items that match the search
220
221      DisplayContents()
222
223      For x = 0 To fileContents.Items.Count - 1 'go through whole list
224          Dim currentItem = fileContents.Items.Item(x)
225          For y = 0 To fileContents.Items.Item(0).SubItems.Count - 1 'search all columns of selected item
226
227              If LCase(currentItem.SubItems.Item(y).Text).Contains(searchFor) Then 'if searchFor is within the text in question
228                  If tempList.Contains(currentItem) = False Then 'if item not already in list(prevents duplicates)
229                      tempList.Add(currentItem) 'add item to list
230                  End If
231              End If
232
233          Next
234      Next
235
236      fileContents.Items.Clear()
237
238          For x = 0 To tempList.Count - 1
239          fileContents.Items.Add(tempList(x)) 'add items from the list into the listview
240      Next
241
242  End Sub
```

Above is the code responsible for the search, which can be found in the ManageFiles form on lines 216 - 242.

3	Projected earnings for the week should be automatically generated.	Yes	 The timetable form automatically calculates the projected earnings for the selected week, and displays them using a text box, as shown below.

```
152                  earnings += (job.LP * timetableItem.jobLength / 60) + job.RP
153                  projectedEarnings += (job.LP * timetableItem.jobLength / 60) + job.RP
154                  timetable.DisplayTimetableItem(timetableItem, customer, job, Me)
```

Above is a segment of code from the ViewTimetable class. Every time a past item is added to the timetable, it will run this code to add it's value to the projected and confirmed earnings, and whenever a future item is added, only the projectedEarnings will increment, as shown below.

```
190
191                 projectedEarnings += (job.LP * timetableItem.jobLength / 60) + job.RP
192                 timetable.DisplayTimetableItem(timetableItem, customer, job, Me)
```

4	Information should be automatically checked for errors, such as words for numerical values.	Yes	Applicable validation routines are applied to all data-input fields where necessary. This is well documented and evidenced in the testing section of the code.

The validation functions can be found in the Validation.vb class.

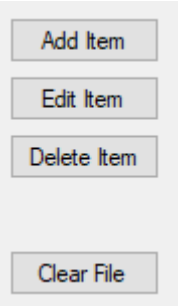
5	All staff should be able to access the required, and only the required, features and functions of the system.	Yes	This feature has been previously discussed and is also evidenced during the testing section of the project.

```
35              If UserList(counter).Username = attemptedID Then 'search for user
36                  If UserList(counter).Password = attemptedPass Then 'check pass of user
37                      validID = True
38                      accessLevel = UserList(counter).AccessLevel
39                  End If
40              End If
41
42              counter += 1
43          End While
44
45          If validID Then
46
47              Dim mainForm = New Main
48              mainForm.Show()
49              My.Application.DoEvents()
50              mainForm.SetAccessLevel(accessLevel, usernameBox.Text)
```

The above code from the Gate form (login screen), fetches the user access level on line 38, and passes it through to the Main form as a parameter on line 50.

```
3       Public Sub SetAccessLevel(accessLevel As String, username As String)
4           If accessLevel = "Admin" Then 'if admin
5               Me.Text += " (Admin)" 'update form title
6           Else
7               'remove/disable admin-only buttons
8               managefilesButton.Dispose()
9               graphButton.Dispose()
10              userNavigateButton.Enabled = False
11          End If
12          greetingLabel.Text += username
13      End Sub
```

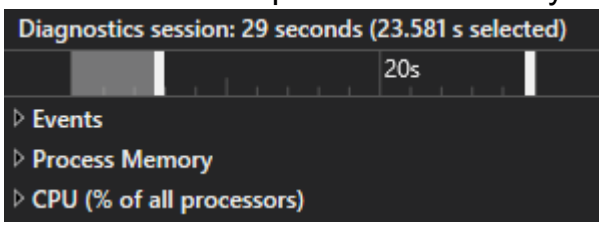
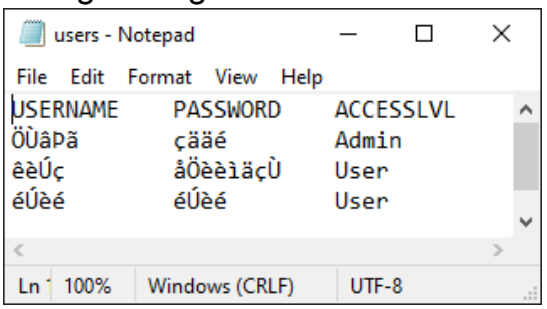
The above code from the Main form is the function that is triggered by the Gate form. This function changes the Main form's contents by

<table>
<tr><td colspan="4">removing/disabling admin-only controls if the user is not of admin status.</td></tr>
<tr><td>6</td><td>The director must be able to create, manage and delete user accounts.</td><td>Yes</td><td>Admin users (such as the director) are able to perform these functions using the following buttons on the Manage Files form:
 
</td></tr>
<tr><td colspan="4">These functions can be seen in the ManageFiles form.

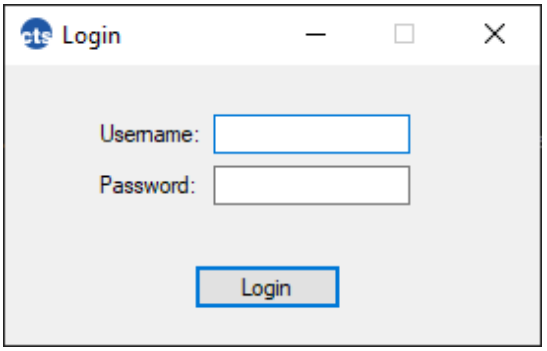
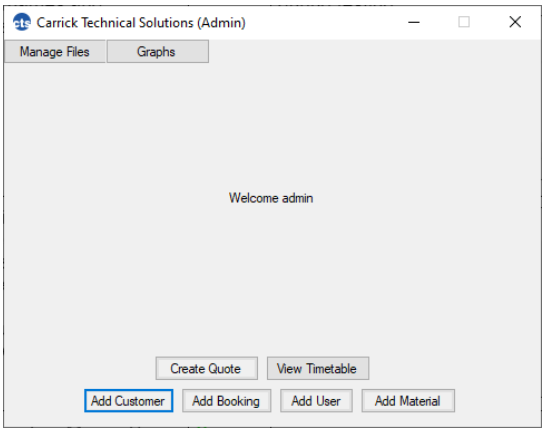
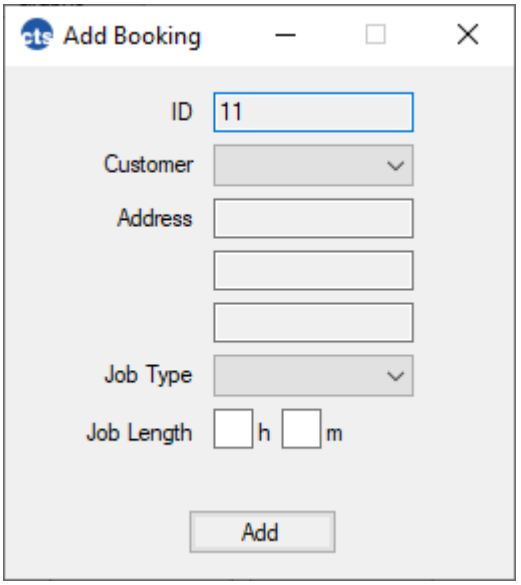
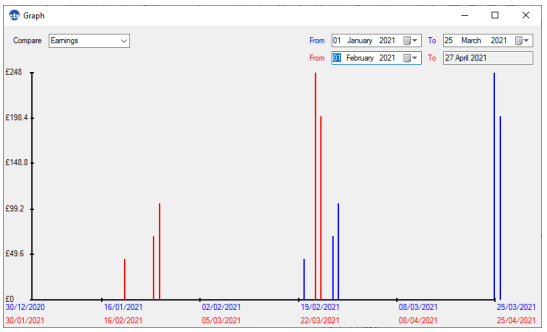
```
124     Private Sub addItemButton_Click(sender As Object, e As EventArgs) Handles addItemButton.Click
125
126         Dim form 'get form from user's selection
127         If filenameBox.Text = "Users.txt" Then
128             form = New AddUser
129
130         ElseIf filenameBox.Text = "Customers.txt" Then
131             form = New AddCustomer
132
133         ElseIf filenameBox.Text = "Bookings.txt" Then
134             form = New AddBooking
135
136         ElseIf filenameBox.Text = "Jobs.txt" Then
137             form = New AddJob
138
139         ElseIf filenameBox.Text = "Materials.txt" Then
140             form = New AddMaterial
141
142         End If
143
144         form.Show() 'open form
145         form.SetParent(Me) 'feeds this form to the new form as a parameter, so the form knows whcih form to alter
146
147     End Sub
```
</td></tr>
</table>

```
149     Private Sub editItemButton_Click(sender As Object, e As EventArgs) Handles editItemButton.Click
150
151         Dim form 'get form from user's selection
152         Dim ID As String
153         If filenameBox.Text = "Users.txt" Then
154             form = New AddUser
155             Dim encryptor As New Encryption 'encrypts the username, so the file record can be found
156             ID = encryptor.Encrypt(fileContents.SelectedItems(0).Text)
157
158             form.Show() 'open form
159             form.SetParent(Me) 'feeds this form to the new form as a parameter, so the form knows whcih form to alter
160             form.EnableEditMode(ID)
161
162         ElseIf filenameBox.Text = "Customers.txt" Then
163             form = New AddCustomer
164
165             Dim currentItem As GlobalVariables.customer 'feeds the selected data into the form so the fields can be filled
166             currentItem.ID = fileContents.SelectedItems(0).Text
167             currentItem.Name = fileContents.SelectedItems(0).SubItems(1).Text
168             currentItem.Number = fileContents.SelectedItems(0).SubItems(2).Text
169             currentItem.Address = fileContents.SelectedItems(0).SubItems(3).Text
170             currentItem.Area = fileContents.SelectedItems(0).SubItems(4).Text
171             currentItem.Postcode = fileContents.SelectedItems(0).SubItems(5).Text
172
173             form.Show()
174             form.SetParent(Me)
175             form.EnableEditMode(currentItem) 'feeds the selected data into the form so the fields can be filled
176
177         ElseIf filenameBox.Text = "Bookings.txt" Then
178             form = New AddBooking
179
180             Dim currentItem As GlobalVariables.booking
181             currentItem.ID = fileContents.SelectedItems(0).Text
182             currentItem.customerID = fileContents.SelectedItems(0).SubItems(1).Text
183             currentItem.jobID = fileContents.SelectedItems(0).SubItems(2).Text
184             currentItem.jobLength = fileContents.SelectedItems(0).SubItems(3).Text
185
186             form.Show()
187             form.SetParent(Me)
188             form.EnableEditMode(currentItem)
189
190         ElseIf filenameBox.Text = "Materials.txt" Then
191             form = New AddMaterial
192
193             Dim currentItem As GlobalVariables.material
194             currentItem.ID = fileContents.SelectedItems(0).Text
195             currentItem.name = fileContents.SelectedItems(0).SubItems(1).Text
196             currentItem.type = fileContents.SelectedItems(0).SubItems(2).Text
197             currentItem.pricePerUnit = fileContents.SelectedItems(0).SubItems(3).Text
198
199             form.Show()
200             form.SetParent(Me)
201             form.EnableEditMode(currentItem)
202
203         End If
204
205     End Sub
```

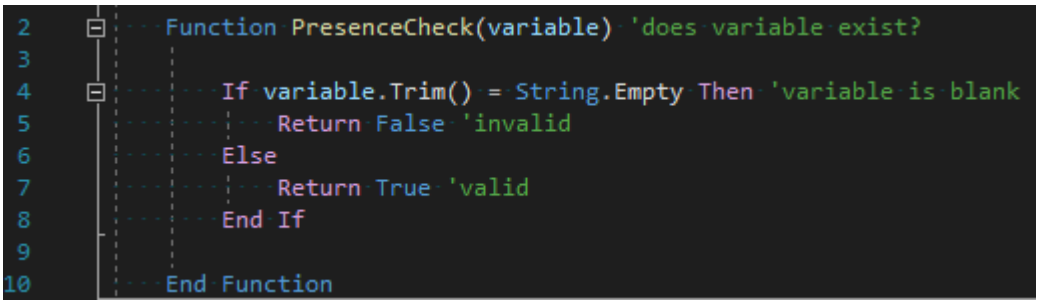
```
268     Private Sub deleteItemButton_Click(sender As Object, e As EventArgs) Handles deleteItemButton.Click
269
270         Dim confirmation As DialogResult 'confirmation prompt
271         confirmation = MessageBox.Show("Are you sure you wish to delete this item? This cannot be undone.", "Confirmation Requrie
272         If confirmation = vbYes Then
273             Dim currentItem = fileContents.Items.IndexOf(fileContents.SelectedItems.Item(0)) + 1 'item to delete
274
275             fileHandler.ClearFile() 'clears file
276             For x = 1 To fileData.Count - 1
277
278                 If x = currentItem Then
279                 Else
280                     fileHandler.WriteLineToFile(fileData(x)) 'adds items back tolist, except for the item to delete
281                 End If
282
283             Next
284
285             fileContents.Items(currentItem - 1).Remove()
286             'DisplayContents() 'OLD
287
288             If filenameBox.Text = "Bookings.txt" Then 'updates timetable if neccesary
289                 Dim timetableCalculator As New TimetableCalculations
290                 timetableCalculator.GenerateTimetable()
291                 MsgBox("Timetable updated.")
292             End If
293
294             editItemButton.Enabled = False 'prevents usingthese buttons when no items are present
295             deleteItemButton.Enabled = False
296         End If
297
298     End Sub
```

7	Users should be able to add entries and information easier.	Yes	This varies depending on several factors, however, on average the digital solution was fasterby an average of 17.8 seconds per item.
The time it takes to perform this task varies depending on the user, and both their level of digital literacy and physical writing speed. From some testing, it took on average 30.4 seconds for a customer to be added to the system (timing from the form opening, to the writing completing), whereas I found that it took an average of 48.2 seconds to record these details manually. Below is an example of one of the system tests:  The code for these functions can be found in the AddBooking, AddCustomer, AddJob, and AddMaterial forms, under the button click event.			
8	All users must have securely encrypted usernames and passwords.	Yes	This has been well documented previously in the evaluation, and during testing. 
The below code is the Encryption class, which is responsible for both encrypting, and decrypting data.			

```vb
Public Class Encryption

    Dim key As Integer = 117

    'encrypts the parmater data using a Caesear Cypher method
    3 references
    Function Encrypt(data As String)

        'split data into an array of numbers
        Dim encryptedData As String = ""
        For x = 0 To data.Count - 1
            Dim charValue = Asc(Trim(Mid(data, x + 1, 1)))

            'shift characters
            If charValue + key <= 255 Then

                charValue += key

            Else

                charValue = key - (255 - charValue)

            End If

            encryptedData += Chr(charValue) 'reconvert to character and place back into string
        Next

        Return encryptedData

    End Function

    'decrypts the parmater data using a Caesear Cypher method
    3 references
    Function Decrypt(data As String)

        'split data into an array of numbers
        Dim encryptedData As String = ""
        For x = 0 To data.Count - 1
            Dim charValue = Asc(Trim(Mid(data, x + 1, 1)))

            'shift characters
            If charValue - key >= 1 Then

                charValue -= key

            Else

                charValue = 255 + (charValue - key)

            End If

            encryptedData += Chr(charValue) 'reconvert to character and place back into string
        Next

        Return encryptedData

    End Function

End Class
```

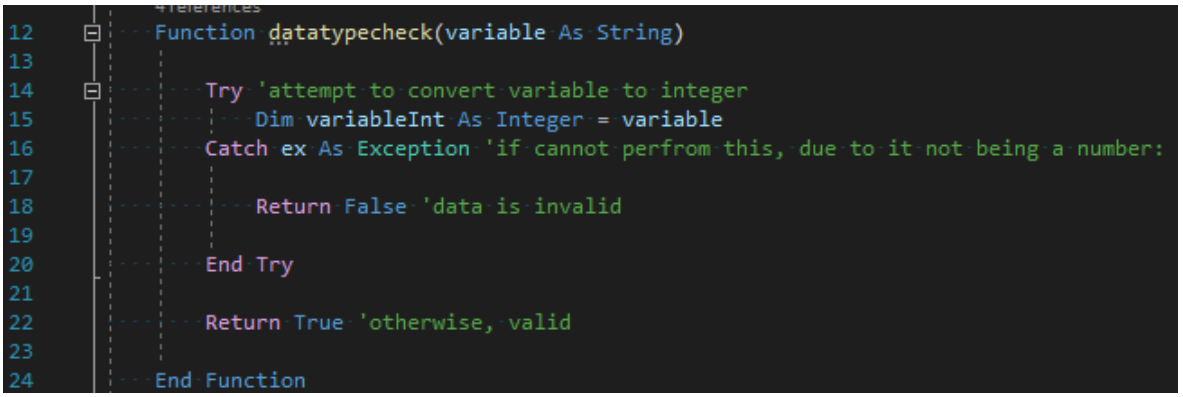
9	Login Screen. Username and password should be typed using a keyboard, and submitted using a mouse to interact with a button.	Yes	
10	Navigation Menu. The user should navigate to the required menu using the mouse to select the corresponding button.	Yes	
11	New Booking. The customer's details (name, address, postcode, etc.) should be entered using a keyboard. Some details, such as 'type of job' should be selected from a dropdown menu using the mouse.	Yes	
12	Financial Menu. The director should be able to select dates and filters through the use of the mouse.	Yes	

13	Presence checks should be used on almost all data inputs (barring optional inputs, and inputs where there is a default value, so it cannot be blank).	Yes	See testing section.



```
Function PresenceCheck(variable) 'does variable exist?

    If variable.Trim() = String.Empty Then 'variable is blank
        Return False 'invalid
    Else
        Return True 'valid
    End If

End Function
```

14	Data type checks should be used on all numerical values that are needed for calculations, such as quantities, prices, etc,	Yes	See testing section.



```
4 references
Function datatypecheck(variable As String)

    Try 'attempt to convert variable to integer
        Dim variableInt As Integer = variable
    Catch ex As Exception 'if cannot perfrom this, due to it not being a number:

        Return False 'data is invalid

    End Try

    Return True 'otherwise, valid

End Function
```

15	Format checks should be used on patterned values, for example postcodes should match the AA000AA format, and phone numbers should match the 07xxxxxxxx format.	Yes	See testing section.

```
Function PostcodeFormat(variable As String)

    If ExactLengthCheck(variable, 7) Then 'check data is 7 characters
        For x = 0 To 1 '1st and 2nd characters
            If RangeCheck(Asc(variable(x)), 65, 90) Then 'between A and Z? (not a - Z !!!)
            Else
                Return False 'invalid data
            End If
        Next

        If datatypecheck(LSet(Mid(variable, 3), 3)) Then 'check that the '000' section of AA000AAA is indeed numerical
        Else
            Return False 'invalid data
        End If

        For x = 5 To 6 'check 6th and 7th characters
            If RangeCheck(Asc(variable(x)), 65, 90) Then
            Else
                Return False 'invalid data
            End If
        Next
    Else
        Return False 'invalid data
    End If

    Return True 'valid data

End Function
```

16	Length checks should be used on all file-stored inputs, to ensure that the entries fit into the records.	Yes	See testing section.

```
Function LengthCheck(variable As String, maxLength As Integer) 'is variable small enough to fit into file?

    If variable.Length <= maxLength Then 'if variable shorter than / same length as limit
        Return True 'valid
    Else
        Return False 'invalid
    End If

End Function
```

17	Two-pass verifications should be used when creating new users, to ensure that the correct passwords are entered.	Yes	See testing section.

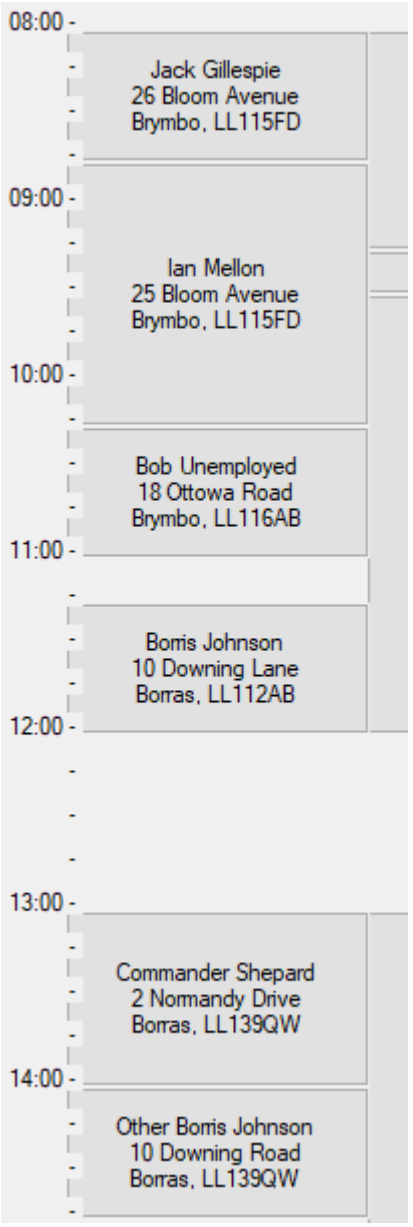
```
Function Verifiy(data, compareTo) 'are parameters the same?

    If data = compareTo Then 'are parameters same
        Return True 'valid data
    Else
        Return False 'invalid data
    End If

End Function
```

18	Exclusive checks should be carried out when new users are created to ensure that	Yes	See testing section.

	the username in question is not already taken.		

```
129             'check if username is unique
130             If inEditMode = False Then 'do not perform this action in edit mode, as when editing items the IDs SHOULD match
131                 For x = 0 To userList.Count() - 1 'search list of users
132                     Dim currentUsername = Trim(userData.Username)
133                     If userList(x).Username = currentUsername Then 'compare IDs, if usernames match
134                         'error message
135                         usernameLabel.ForeColor = vbRed
136                         errorMessage = "Username is taken."
137                         allIsValid = False 'invalid data
138 
139                     End If
140                 Next
141             End If
```

From AddUser form.

19	Timetable should be generated to ensure that customers are ordered based on their geographical proximity.	Yes	 This image shows that the algorithm has correctly sorted the

			bookings by their location, as the first group of items are all Brymbo addresses, with the second group being Borras. As well as this, the items are sorted within their area groups by postcode, which can be seen by the fact that bookings with the same postcodes are next to each other.
The timetable generation is performed by the TimetableCalculations.vb class, which is called by the AddBooking form on line 117, and the ManageFiles form on line 290.			
20	Quote prices should be calculated from the input data.	Yes	 The quote calculation job successfully calculates the hourly rate and resource costs of quotes.

```
32      'ADDS THE MATERIAL FROM THE SLSECT MATERIAL FORM TO THIS ONE
        1 reference
33      Public Sub AddMaterialToList(material As GlobalVariables.material)
34
35          Dim newMaterial = materialsList.Items.Add(material.name) 'adds material to list
36          newMaterial.SubItems.Add(material.pricePerUnit)
37          newMaterial.SubItems.Add(material.type)
38
39          'enables buttons
40          removeMaterialButton.Enabled = True
41          clearButton.Enabled = True
42
43          'maths
44          If material.type = "Resource" Then
45              resourcePrice += material.pricePerUnit
46              resourcePriceBox.Text = resourcePrice
47          Else
48              labourPrice += material.pricePerUnit
49              labourPriceBox.Text = labourPrice
50          End If
51
52      End Sub
```

The above code from the AddJob form shows how the resource and labour prices are incremented by the amount of the new material, on lines 43 to 50.

```
54      'REMOVES THE CURRENT SELECTED MATERIAL
        0 references
55      Private Sub removeMaterialButton_Click(sender As Object, e As EventArgs) Handles removeMaterialButton.Click
56
57          If materialsList.SelectedItems().Item(0).SubItems(2).Text = "Resource" Then
58              resourcePrice -= materialsList.SelectedItems().Item(0).SubItems(1).Text
59              resourcePriceBox.Text = resourcePrice
60          Else
61              labourPrice -= materialsList.SelectedItems().Item(0).SubItems(1).Text
62              labourPriceBox.Text = labourPrice
63          End If
64
65          materialsList.Items.Remove(materialsList.SelectedItems().Item(0)) 'removes item from listview
66
67          If materialsList.Items.Count = 0 Then 'if all items have been removed
68              removeMaterialButton.Enabled = False 'disable the buttons, as no items can be selected
69              clearButton.Enabled = False
70          End If
71
72      End Sub
```

The above code is the deletion process from the same class, and has the opposite effect, decrementing the price values as an item is removed.

21	New user data should be encrypted.	Yes	See objective 8.
22	Projected earnings from the week's jobs should be calculated.	Yes	See objective 3.
23	Past earnings should be comparable.	Yes	 The timetable form automatically calculates the confirmed earnings for the selected past week, and displays them using a text box, as shown above. Past earnings can also be compared directly using the Graph form.
24	Timetable should be viewable by all staff, as a visual representation.	Yes	Regardless of whether the user is an admin or standard user, they are able to access the timetable via the main form.

```
41      Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
42
43          ViewTimetable.Show()
44
45      End Sub
```

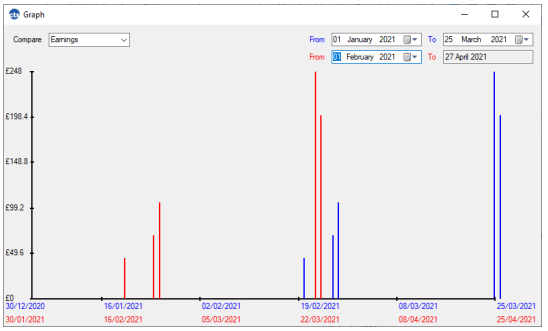
The above code shows that the ViewTimetable from can be accessed using the button on the Main form.

The function DisplayTimetable() from the ViewTimetable form (line 104 to 292) displays the timetable, using the Timetable.vb and TimetimeButton.vb controls, and instantiating them. This function handles the calculations for the timetable, however the application of these values are applied within the TimetableButton.vb control, as shown below.

```
Sub SetButtonProperties(text As String, size As Size, position As Point, bookingData As GlobalVariables.timetableItem, cust

    'object properties
    Me.Size = size
    button.Size = size

    Me.Location = position

    button.Text = text

    If customerData.Name = "Not Found" Then
        button.Text = "Not Found"
        button.ForeColor = Color.Red
        button.Enabled = False
        Return
    End If

    'data for click event
    Me.bookingData = bookingData
    Me.customerData = customerData
    Me.jobData = jobData

    Dim currentWeek As String = Now.Date
    If bookingData.week <> "" Then
        If bookingData.week <= currentWeek Then
            button.ForeColor = Color.Gray
        End If
    End If

End Sub
```

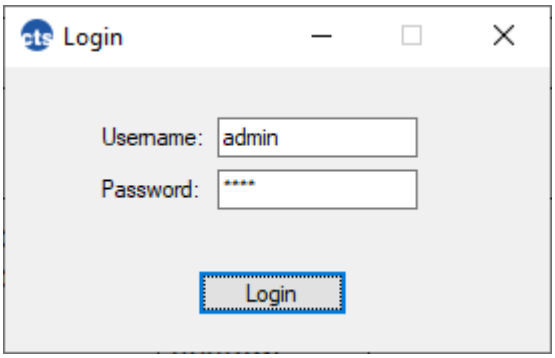
25	Timetable should be printable.	No	This objective was not achieved, due to time constraints. I spoke to the client and explained that due to complications it may be necessary to not implement this feature at release in order to meet the deadline. I was assured that this was okay, as he can still create hard copies of timetables by taking screenshots, and printing those, and so losing this feature was not a major loss. It is planned to add this feature in a future update.
26	Quotes should be viewable by the director. Results of calculations should be viewable by the director.	Yes	This can be achieved using the manage files form and selecting ‘Jobs.txt’.

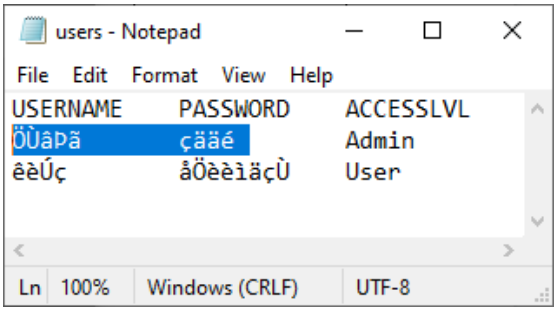
This objective (and objective 27 below) are both achieved using the ManageFile form.

27	A list of users should be viewable by the director.	Yes	This can be achieved using the manage files form and selecting ‘Users.txt’.
28	A graph should display the earnings of days between two sets of dates, where they can be directly compared.	Yes	
This is achieved using the Graph form, using its GetValuesForGraph() and GenerateGraph() functions, on lines 88 - 161 and 164 - 261 respectively.			

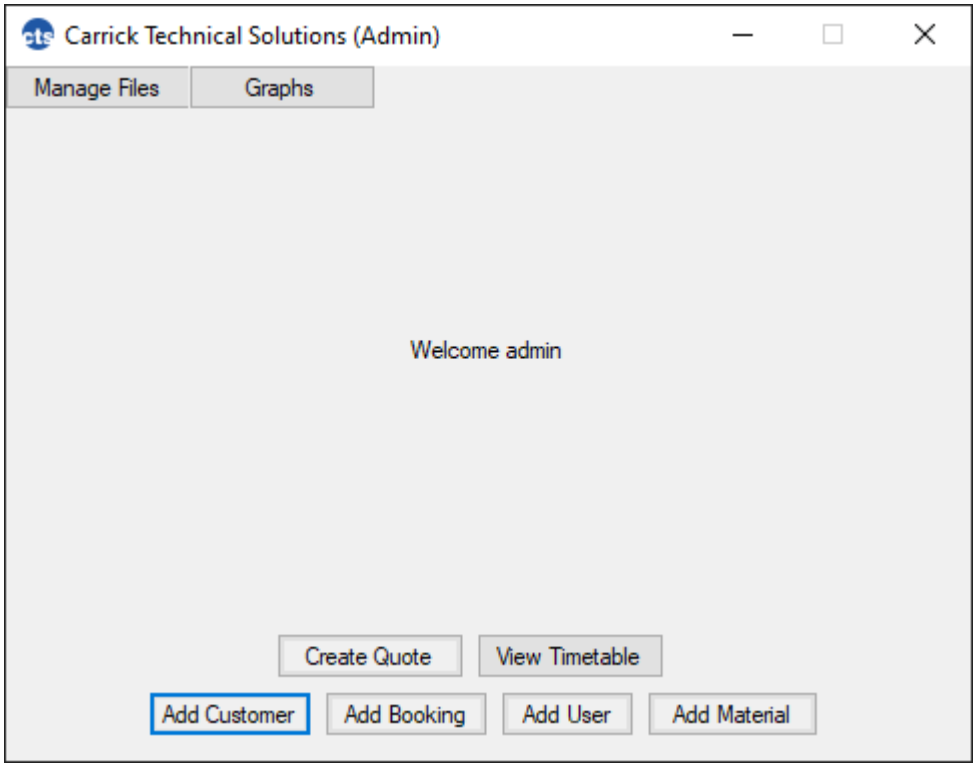
Testing

Form: **Login**					
Test No.	**Operation Tested**	**Data**	**Purpose**	**Expected Outcome**	**Observed Outcome**
1	Login with correct username and password	admin root	To allow users to access the system	Open Main form	**Main form opened**



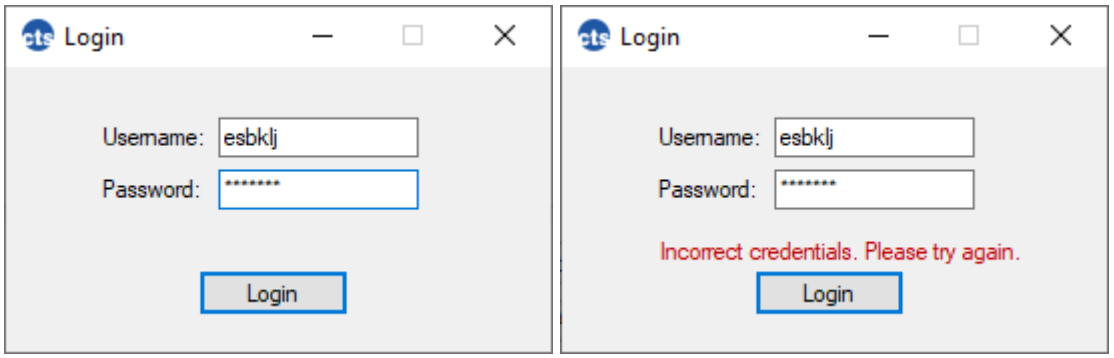


ÖÙâÞã and çääé are the encrypted forms of ‘admin’ and ‘root’ (encryption tested later on).



These screenshots show that when the details ‘admin’ and ‘root’ are entered, the program has found the corresponding details in the users.txt file, accepted them as correct, and so opened the Main form.

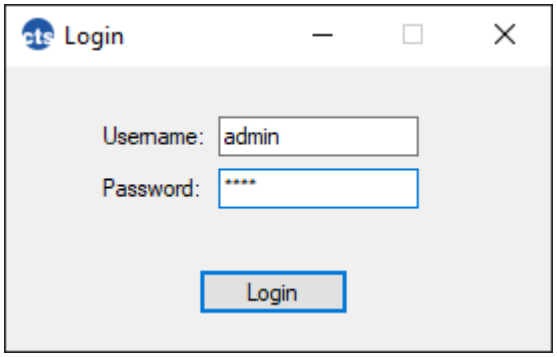
2	Login with incorrect username and password	esbklj skejbfg	To prevent unauthorised access to the system	Error message	**Error message**

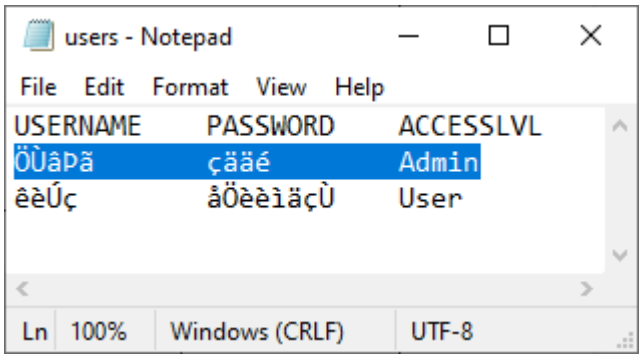


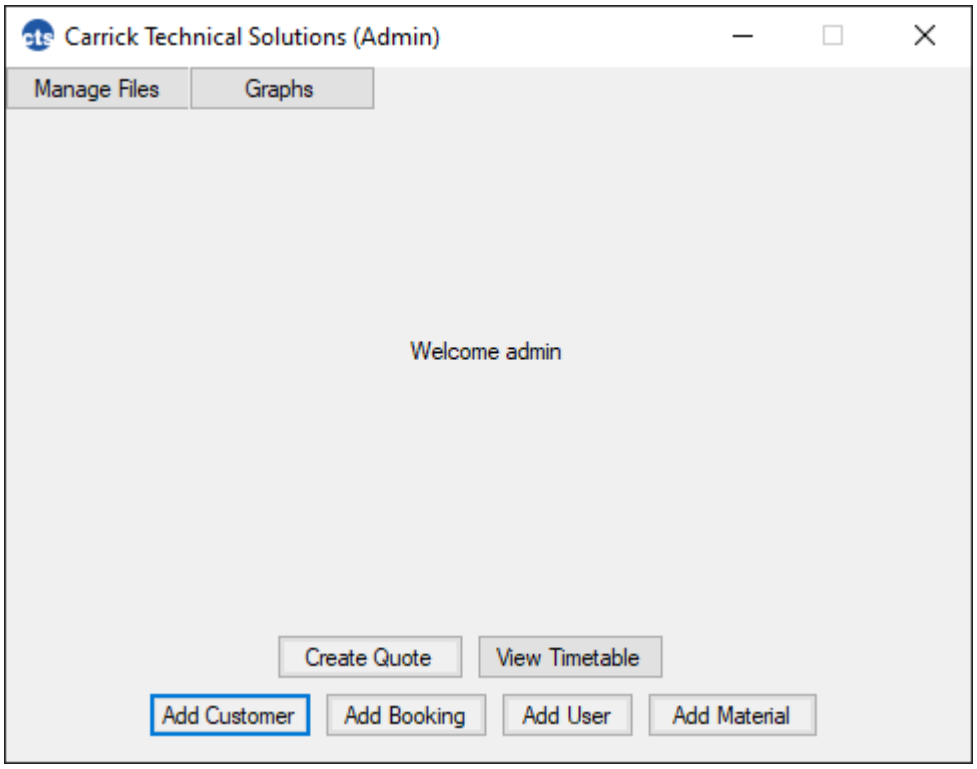
These screenshots show that the system does not open the Main form regardless of the data, as it refuses to open the Main form when incorrect data, such as that in the screenshots above, is entered.

3	Login as admin	admin root	To ensure administrators have	Open Main form as admin	**Opened Main form as admin**

			access to the correct options		



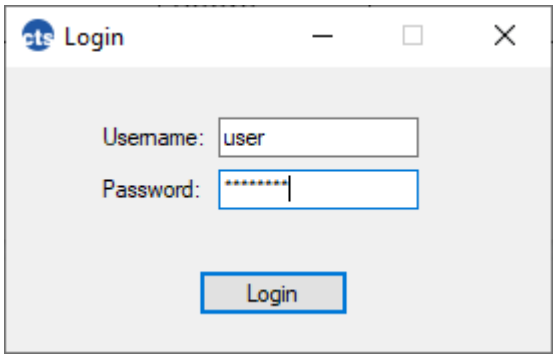


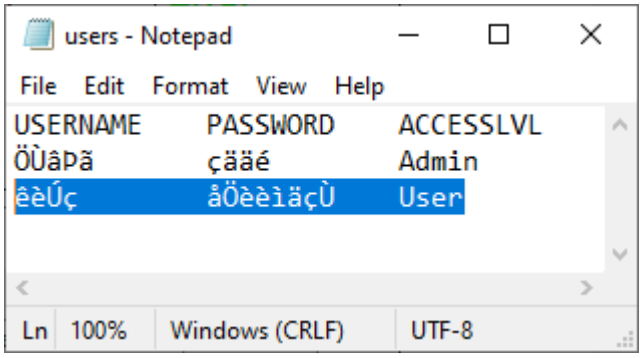


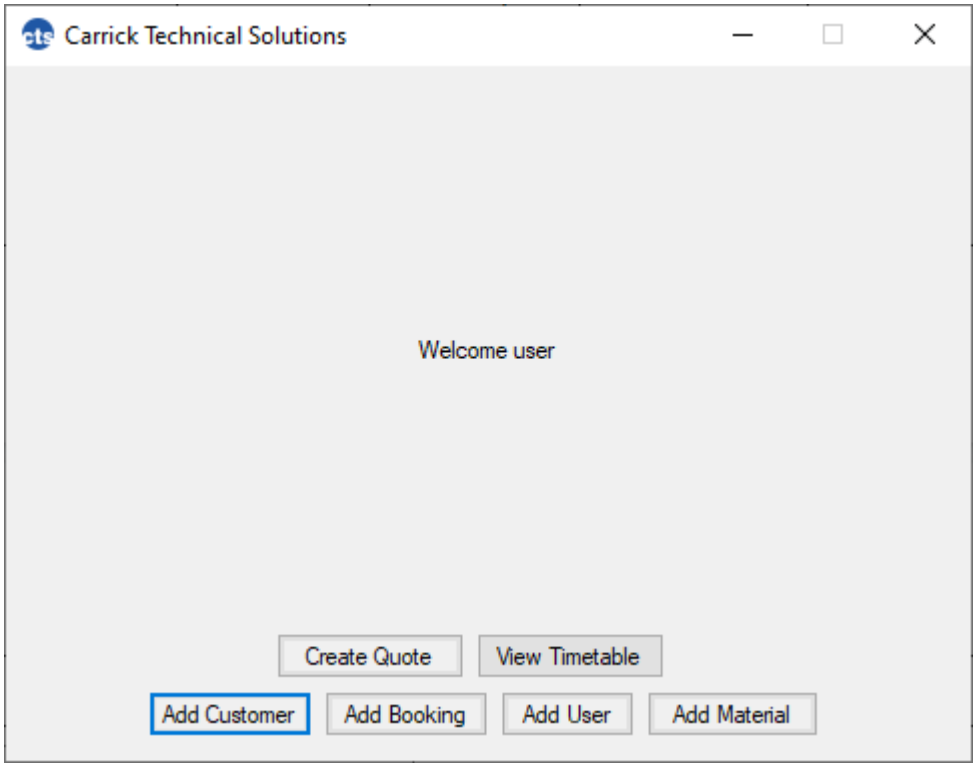
As the user is an admin, the ’Manage Files’ and ‘Graphs’ buttons are available. As well as this, “(Admin)” is in the header. This means admins are able to access all the features of the system.

4	Login as	user	To ensure	Open Main	**Main form**

	standard user	password	only administrators have access to the full options, and standard users cannot.	form as user	**opened as user**

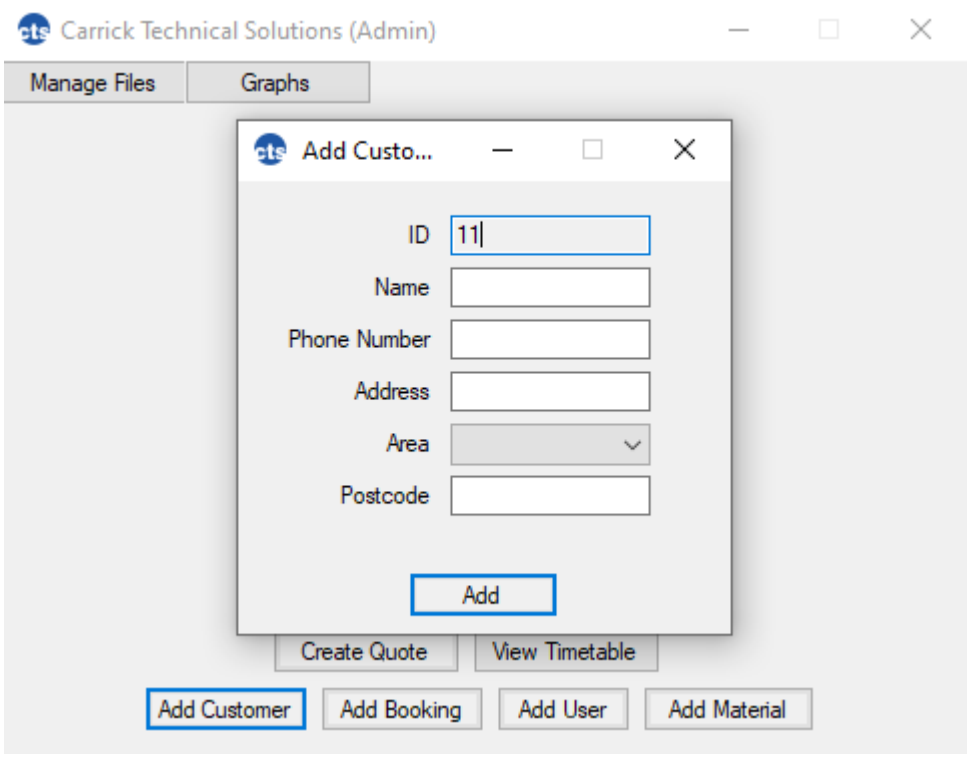




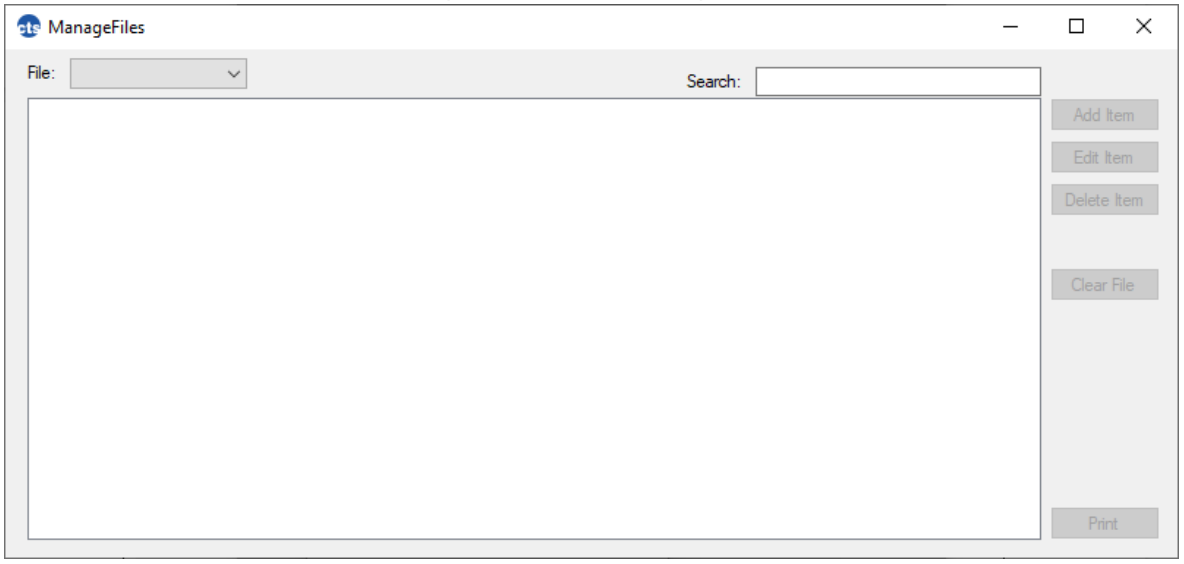


As the user is not an admin, the 'Manage Files' and 'Graphs' buttons are not available. This means that only admins are able to access all the features of the system, and users are limited to data entry, quote calculation, and timetable viewing.

Form: **Main**					
Test No.	**Operation Tested**	**Event**	**Purpose**	**Expected Outcome**	**Observed Outcome**
1	Use navigation buttons to access other forms	Click on a navigation button	To allow users to access the features of the system	Open respective form	**Main form opened**

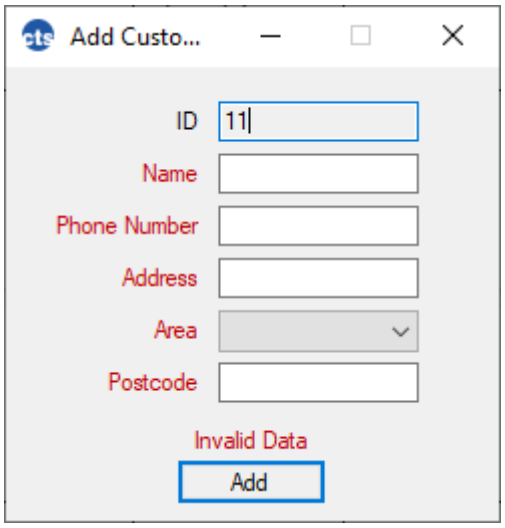


This screenshot demonstrates that using the 'Add Customer' button opens the correct form that it relates to. All other buttons were tested too, including the admin-exclusive buttons (as shown below).



Validation Class,(using **Add Customer**, **Add Material**, and **Add User** form)					
Test No.	**Operation Tested**	**Data**	**Purpose**	**Expected Outcome**	**Observed Outcome**
1	Presence Check		This check should prevent vital fields from being left	Error message	**'Invalid Data' message**

	Invalid data		blank		

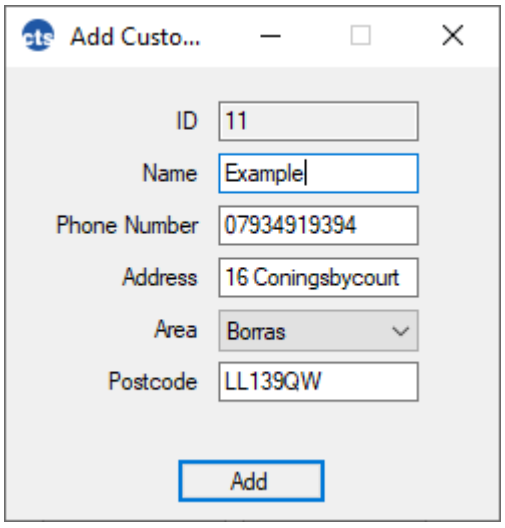


As this screenshot shows, leaving fields blank prevents the form from accepting the data and proceeding to store the data.

```
Function PresenceCheck(variable) 'does variable exist?

    If variable.Trim() = String.Empty Then 'variable is blank
        Return False 'invalid
    Else
        Return True 'valid
    End If

End Function
```

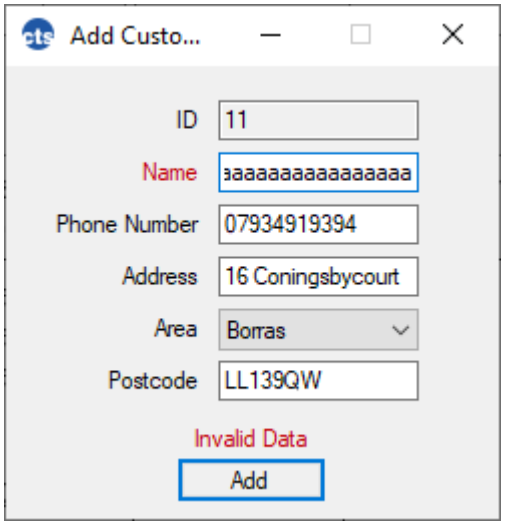
Above is the presence check function.

	Presence Check Valid data	example	This check should prevent vital fields from being left blank	No error message	**No error message**



2	Length	aaaaaaaa	This check should	Error	**‘Invalid**

	Check Invalid data	aaaaaaaaaaaaaaaaaa (a*26)	prevent the user from entering data longer than a predefined length limit.This is as the files are fixed length, and so entering data longer than this limit would result in the truncation of characters.	message	**Data' message**



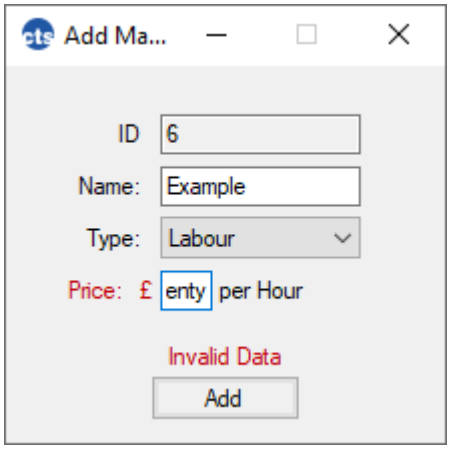
This screenshot shows that as the data (26 characters) is longer than the variable limit (25 characters, in this instance) the check will fail, and prevent the form from proceeding to store the data.

```
Function LengthCheck(variable As String, maxLength As Integer) 'is variable small enough to fit into file?

    If variable.Length <= maxLength Then 'if variable shorter than / same length as limit
        Return True 'valid
    Else
        Return False 'invalid
    End If

End Function
```

Above is the length check function from the validation class.

3	Data Type check Invalid data	seventy	This check should prevent the user from entering non-numerical values into fields which require numerical data, as this data will be used in	Error message	**'Invalid Data' message**

			calculations, and so non-numerical values could break.		



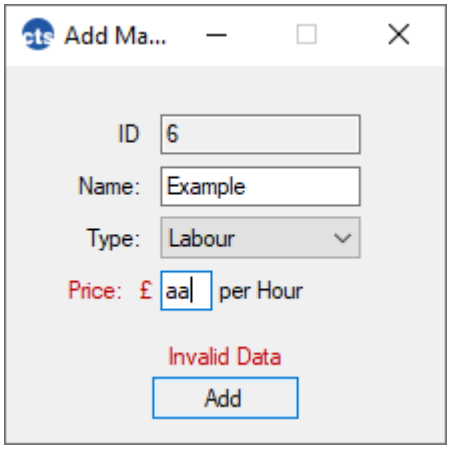
This screenshot shows that the data has failed its validation checks, and so the form cannot proceed to store the data. **However**, this field also uses a length check, with a limit of 3 characters, and the data ‘seventy’ exceeds this. As a consequence of this, the error may not be the result of the data type check, and may just be caused by the length check.

```
106         'VALIDATE REQUIRED FIELDS
107         For x = 0 To (toValidate.Length / 2) - 1 'for each field
108             Dim field = toValidate(0, x), dataIsValid = True
109 
110             'presence check
111             dataIsValid = validation.PresenceCheck(field.Text) And dataIsValid
112 
113             'length check
114             dataIsValid = validation.LengthCheck(field.Text, toValidate(1, x)) And dataIsValid
115 
116             If dataIsValid Then
117                 GetNextControl(field, False).ForeColor = vbDefault
118             Else
119                 GetNextControl(field, False).ForeColor = vbRed 'highlight corresponding label
120             End If
121 
122             allIsValid = dataIsValid And allIsValid
123         Next
124 
125         'data check
126         If validation.datatypecheck(priceBox.Text) Then
127 
128             'range check
129             If validation.RangeCheck(priceBox.Text, 1, 999) Then
130                 allIsValid = allIsValid And True
131                 priceLabel.ForeColor = vbDefault
132             Else
133                 allIsValid = False
134                 priceLabel.ForeColor = vbRed
135             End If
136 
137         Else
138             allIsValid = False
139             priceLabel.ForeColor = vbRed
140         End If
141 
142         If allIsValid Then
143             addButton.Cursor = Cursors.Default
144             errorLabel.Text = ""
145         Else
146             addButton.Cursor = Cursors.No 'feedback to user
147             errorLabel.Text = "Invalid Data"
148         End If
```

This code shows that the program runs a presence and length check on every field, and then a data type and range check on the price field. These checks are done using the Validation class (in this case instantiated as an object called ‘validation’). This shows that if the data passes the data type check, it can still fail the length check, and as a result of this, the data ‘seveny’ will fail the length check, and so the error message seen in the above screenshots may not be caused due to the data type check, and so cannot be used as proof that it works.

3.1	Data Type check	aa	This check should prevent the user from entering non-numerical values into fields which require numerical data, as this data will be used in	Error message	**‘Invalid Data’ message**

			calculations, and so non-numerical values could break.		

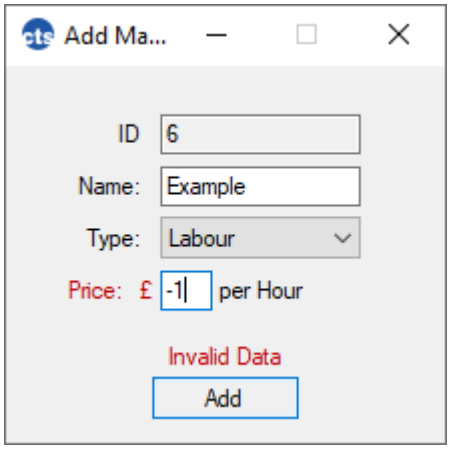


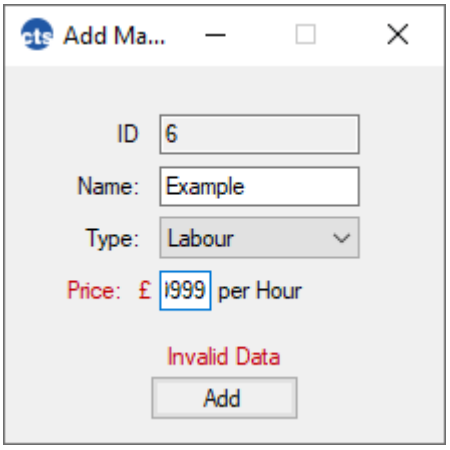
This screenshot shows that the data has failed its validation checks, and so the form cannot proceed to store the data. This test is conclusive proof that the data type check works, as 'aa' is short enough to pass the length check, and no other checks are used.

```
12  Function datatypecheck(variable As String)
13
14      Try 'attempt to convert variable to integer
15          Dim variableInt As Integer = variable
16      Catch ex As Exception 'if cannot perfrom this, due to it not being a number:
17
18          Return False 'data is invalid
19
20      End Try
21
22      Return True 'otherwise, valid
23
24  End Function
```

Above is the code of the data type function, showing how it works.

4	Range check Invalid	-1, 9999	This check should prevent numerical data from being outside of a predetermined range. This prevents impossible values (such as negative time), or values which can break calculations (such as dividing by zero)	Error message	**'Invalid Data' message**





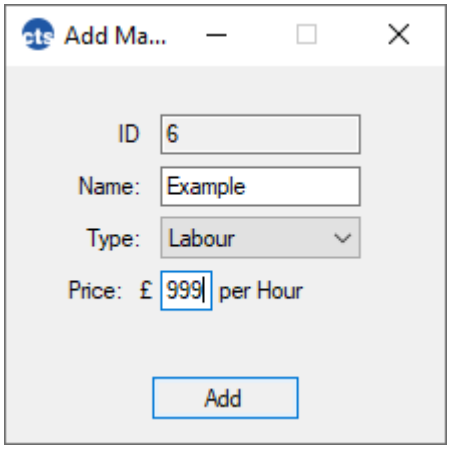
These screenshots show that both the values of -1 and 9999 are invalid data, and so rejected by the system. This shows that data cannot be smaller or greater than the predefined limits (in this case 1 <= x <= 999).

```
46  Function RangeCheck(variable, minBound, maxBound) 'is variable within bounds?
47
48      Dim value
49      value = Convert.ToDouble(variable) 'convert value to number
50
51      If value < minBound Or value > maxBound Then 'is value outside of accepted bounds?
52          Return False 'invalid data
53      Else
54          Return True 'valid data
55      End If
56
57  End Function
```

Above is the range check function from the validation class, showing how it works. As this function converts data to a number, a data type check is required **before** this check, as if value is a string line 49 will create runtime errors.

The values of minBound and maxBound are predefined and unique to each numerical field.

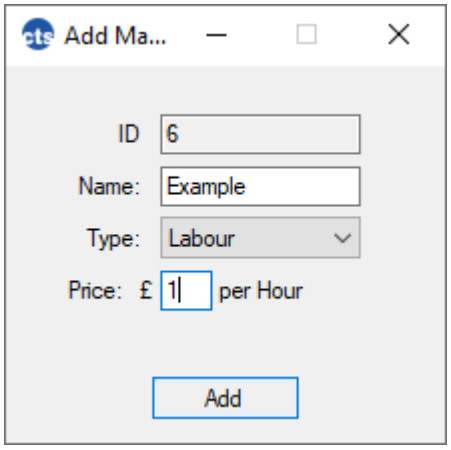
5	Range check Extreme high		This check should prevent numerical data from being outside of a predetermined range. This prevents impossible values (such as negative time), or values which can break calculations (such as dividing by zero)	No error message	**No error message**



This screenshot shows that the absolute value of 999 is accepted as valid.

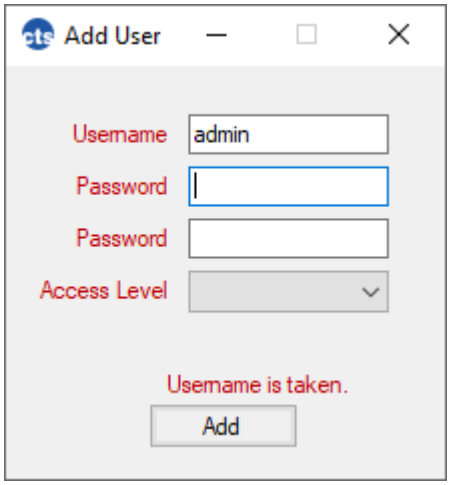
This also proves the data type check works for valid data, as range checks are only performed if the data passes the data type check, as only numerical values can be used in a range check.

6	Range check Extreme low	1	This check should prevent numerical data from being outside of a predetermined range. This prevents impossible values (such as negative time), or values which can break calculations (such as dividing by zero)	No error message	**No error message**

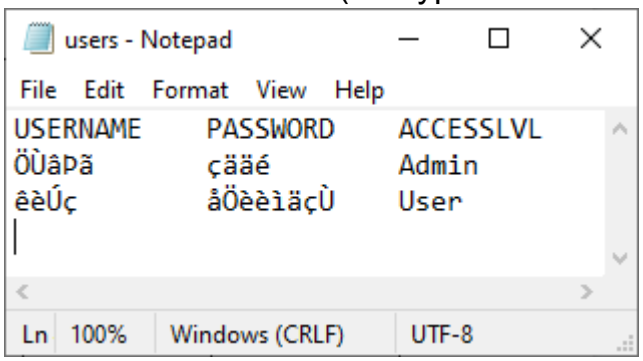


This screenshot shows that the absolute value of 1 is accepted as valid.

This also proves the data type check works for valid data, as range checks are only performed if the data passes the data type check, as only numerical values can be used in a range check.					
7	Unique check	admin	This check should prevent multiple data entries from sharing the same primary key, as this would prevent data normalisation, and lead to complications when handling the data.	Error message	**‘Username is taken’ message**



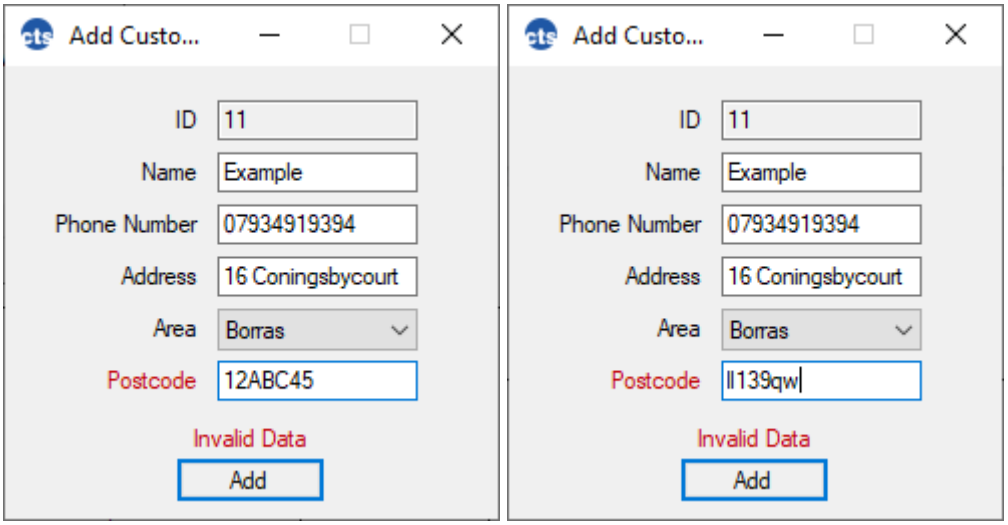
This screenshot shows that the system will not allow multiple users to have the same username (primary key of the users.txt file). As the text file already has the user ‘ÖÙâÞã’ (encrypted from of ‘admin’).



```
129             'check if username is unique
130             If inEditMode = False Then 'do not perform this action in edit mode, as when editing items the IDs SHOULD match
131                 For x = 0 To userList.Count() - 1 'search list of users
132                     Dim currentUsername = Trim(userData.Username)
133                     If userList(x).Username = currentUsername Then 'compare IDs, if usernames match
134                         'error message
135                         usernameLabel.ForeColor = vbRed
136                         errorMessage = "Username is taken."
137                         allIsValid = False 'invalid data
138 
139                     End If
140                 Next
141             End If
```

Above is the code that performs this check. As this is the only instance where the primary key is manually entered (all other primary keys are generated by the system, and so cannot be the same) this function is only needed once, and so is in the **Add User** form, and not the **Validation** class.

8	Format check: Postcode Invalid data	12ABC45, ll139qw	This check will ensure that all postcodes are in the AA000AA format, as all valid UK postcodes are. This prevents the entry of incorrect information.	Error message	**‘Invalid data’ message**

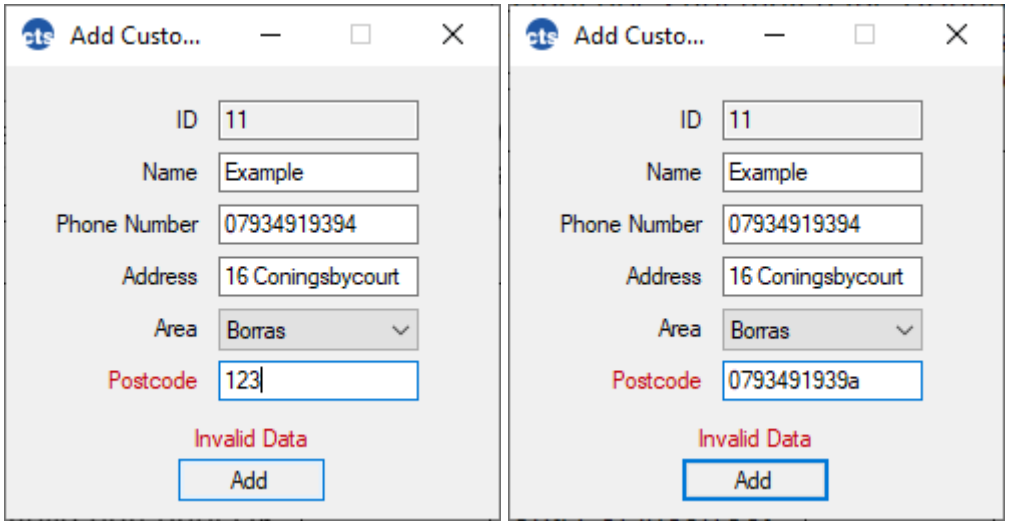


These screenshots show that data that does not match the AA000AA format is invalid and not allowed by the system. The second screenshot shows that aa000aa is also not accepted, and that data entered must be in a capitalised form.

```
Function PostcodeFormat(variable As String)

    If ExactLengthCheck(variable, 7) Then 'check data is 7 characters
        For x = 0 To 1 '1st and 2nd characters
            If RangeCheck(Asc(variable(x)), 65, 90) Then 'between A and Z? (not a - Z !!!)
            Else
                Return False 'invalid data
            End If
        Next

        If datatypecheck(LSet(Mid(variable, 3), 3)) Then 'check that the '000' section of AA000AAA is indeed numerical
        Else
            Return False 'invalid data
        End If

        For x = 5 To 6 'check 6th and 7th characters
            If RangeCheck(Asc(variable(x)), 65, 90) Then
            Else
                Return False 'invalid data
            End If
        Next
    Else
        Return False 'invalid data
    End If

    Return True 'valid data

End Function
```

The above code is the format check for postcodes, and is actually composed of exact length checks, range checks and a data type check -- however it only tests specific parts of the data, and doesn't perform them on the whole string. All of these validations have been tested above, and so this test is just to ensure they are utilized correctly.

9	Format check: Phone number Invalid data	123, 0793491939a	This check will ensure that all phone numbers are numerical,and 11 digits, as all valid personal UK numbers are. This prevents the entry of incorrect information.	Error message	**'Invalid data' message**



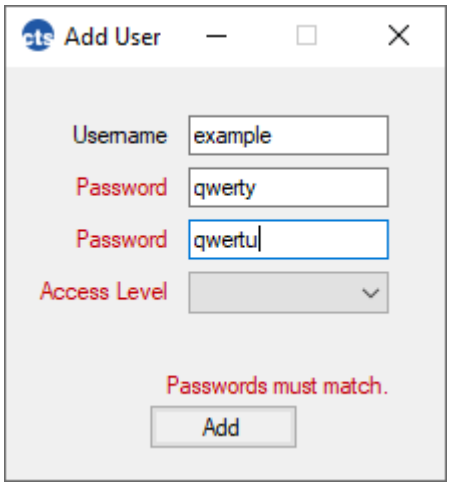
These screenshots show that if data is non-numerical (screenshot 2) or not

11 digits (screenshot 1) then it is marked as invalid and not allowed.

```
Function PhoneNumberFormatCheck(variable As String)

    Try 'check data is numerical
        'this is done by diving the variable by itself, which should result in 1, however for non-numerical values will trigger an error
        'a normal data type check cannot be used as if the phone number begins with a 0, e.g. (07934...) it cannot be stores as an integer
        'and so will trigger as invalid, even though it should be valid
        Dim check As Integer = variable / variable

        Return ExactLengthCheck(variable, 11) 'valid if data also correct length, invalid if not

    Catch ex As Exception
        Return False 'invalid
    End Try

End Function
```

This is the code for the phone number format check. It checks the value is both numerical, and exactly 11 digits. To check if the data is numerical, the data type check is **not** used, instead its own unique test is performed. This is done by dividing the variable by itself, which should result in 1, however for non-numerical values it will trigger an error. The normal conversion data type check cannot be used as if the phone number begins with a 0, e.g. (07934...) it cannot be stored as an integer, as integers cannot be 0... and so will trigger as invalid, even though it should be valid. The division test is used instead to overcome this.

10	Double input verification Invalid data	qwerty qwertu	This verification check forces the user to input certain data twice, to reduce the likelihood of errors and mistakes due to mistyping.	Error message	

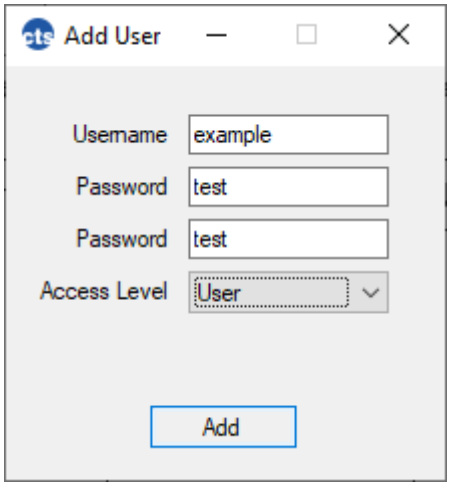


This screenshot shows that the data has been disallowed due to the fact that the two passwords do not match.

```
104     Function Verifiy(data, compareTo) 'are parameters the same?
105
106         If data = compareTo Then 'are parameters same
107             Return True 'valid data
108         Else
109             Return False 'invalid data
110         End If
111
112     End Function
```

This code performs the double-check verification, and compares the two fields' data, and will only allow the data to pass as valid if they are the same.

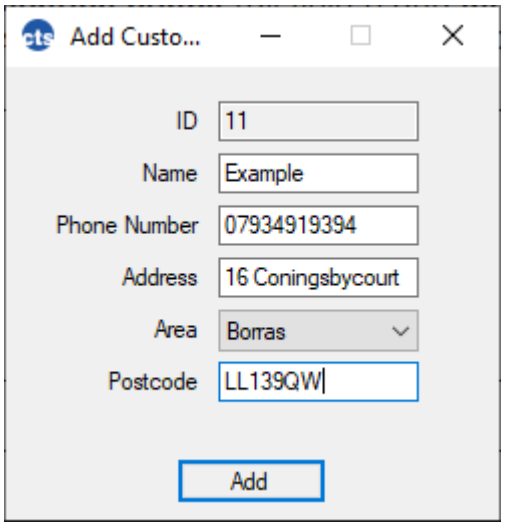
11	Unique check, Double check verification, Presence check, Length check Valid data	example test test User	(See above examples)	No error message	**No error message**



This screenshot shows that the **unique check** (username), **double check verification** (password fields), **presence check** (all fields) and **length check** (username, password fields) does accept valid data, and doesn't just reject all data.

12	Format checks: Postcode and Phone Number valid data	07934919394 LL139QW	(See above examples)	No error message	**No error message**

	Valid data				

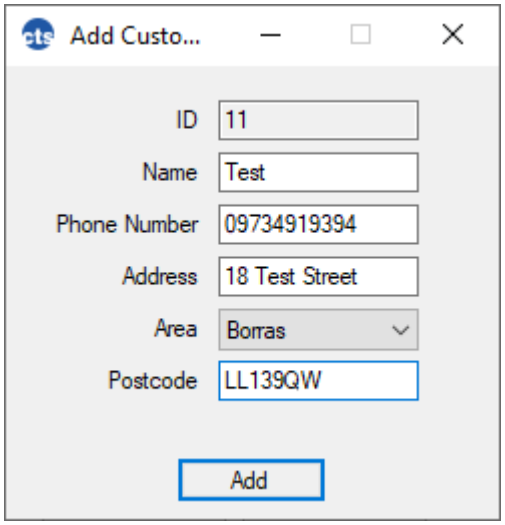


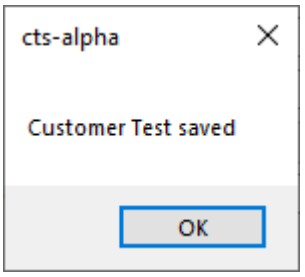
This screenshot shows that the **postcode format check** and **phone number format check** both accept valid data entries.

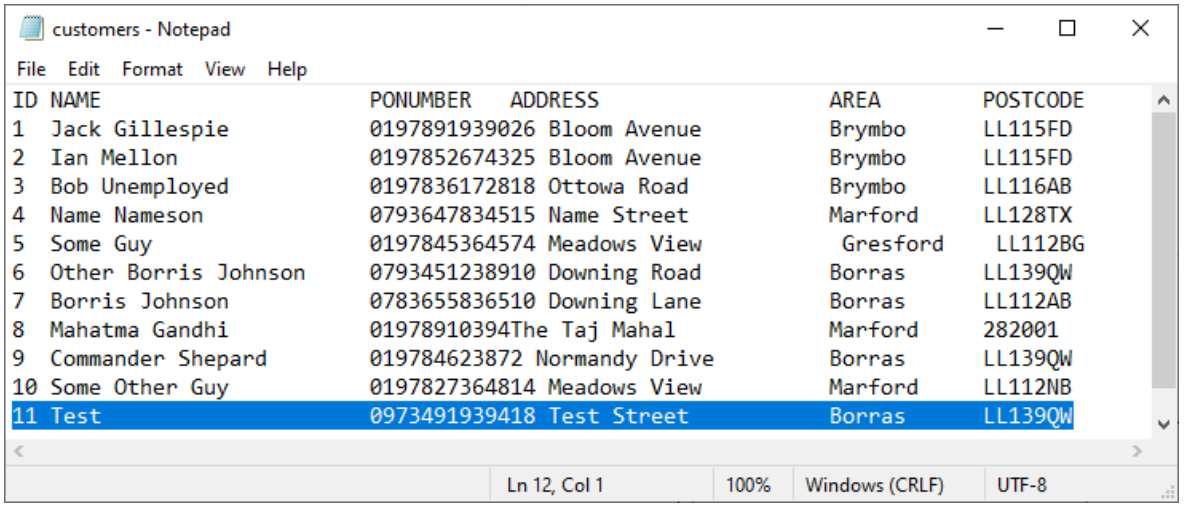
As all data input forms use the **Validation class** to validate, each type only needs to be tested once, on one form, as opposed to testing every field on every form.

Form: **Add Customer**					
Test No.	**Operation Tested**	**Data**	**Purpose**	**Expected Outcome**	**Observed Outcome**
1	Storing input data into file		This will allow the user to store data, which will allow it to be used by the system, even after the system has been closed and reopened, and even after the computer has powered down and backup, as it is storing the data in non-volatile storage. This data	Data stored in file	**'Customer saved' message, and data written correctly to file**

			will be used by several other parts of the system.		



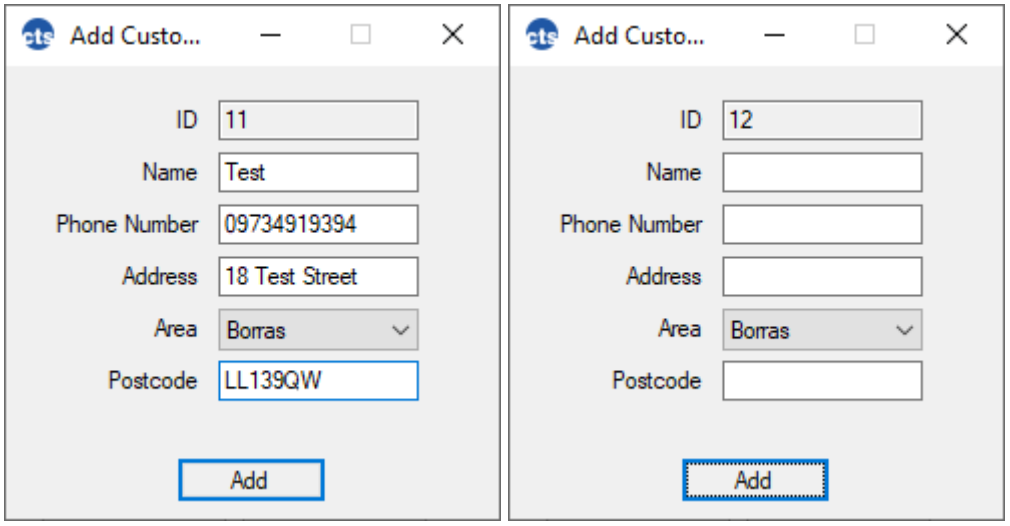




These screenshots show that the data has been successfully stored in the customers.txt file, and the user is informed of this, via an msgBox.

This process uses a shared fileHandling class, and so will perform the same for all data input forms, so only needs testing once.

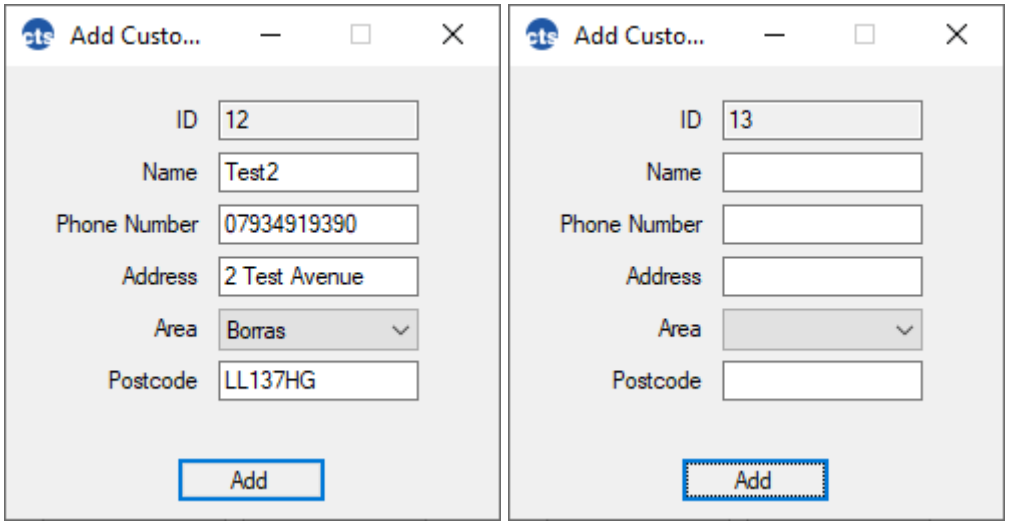
2	ID auto increment	ID: 11	This will automatically update the ID and clear all fields, ready for the next entry.	All files clear. ID should be 12.	**ID updated. Not all fields cleared.**



This screenshot shows that the ID has correctly updated, and all fields have blanked, **except** for the area field.

```
59                'reset fields
60                IDBox.Text = customersFileHandler.GetID()
61                nameBox.Text = ""
62                numberBox.Text = ""
63                addressBox.Text = ""
64                areaBox.Text = ""
65                postcodeBox.Text = ""
```

2.1	^	ID: 12	^	All fields blank	**All fields blank**



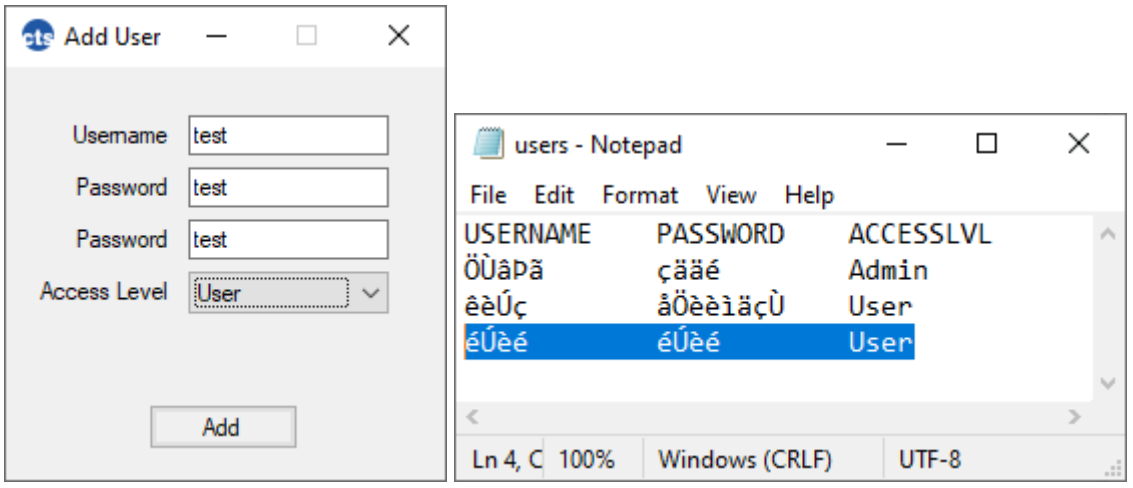
This screenshot shows that the area field has now cleared. This was done by altering line 64 of the code, as shown in the screenshot below, as the areaBox is a combo box and cannot be addressed as a text box.

```
59          'reset fields
60          IDBox.Text = customersFileHandler.GetID()
61          nameBox.Text = ""
62          numberBox.Text = ""
63          addressBox.Text = ""
64          areaBox.SelectedIndex = -1
65          postcodeBox.Text = ""
```

Form: **Add User**

Test No.	Operation Tested	Data	Purpose	Expected Outcome	Observed Outcome
1	Encryption	test	Confidential data (such as usernames and passwords) is encrypted to prevent people from accessing the system by reading the users.txt file and finding out the login details.	éÚèé (see below)	**éÚèé**

The encryption for this system is Caesar cipher, with a shift value of 117, between the bounds of the ASCII values 1 and 255. The word test (116, 101, 115, 116) will shift to (233, 218, 232, 233) éÚèé.

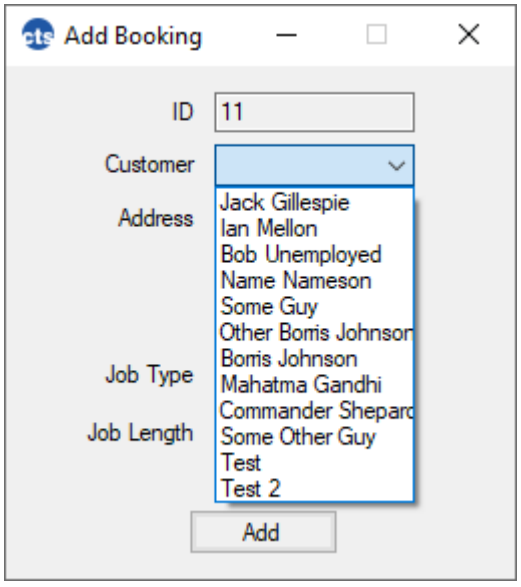


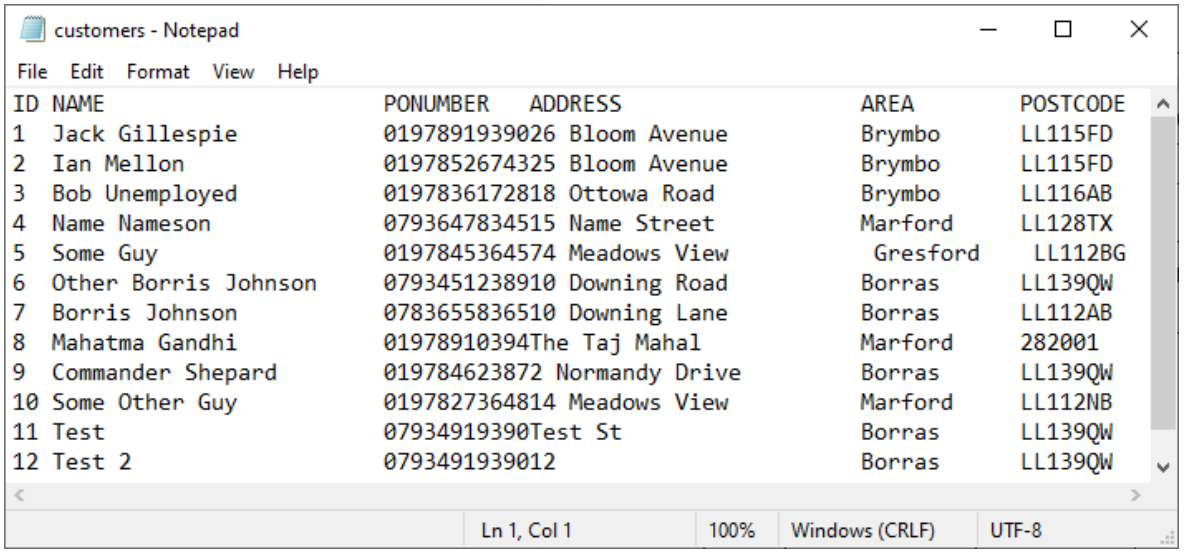
This screenshot shows that ‘test’ has correctly shifted to ‘éÚèé’, which is the same result as the manual calculation, meaning the encryption has worked correctly.

The encryption can also be proven using the decryption feature, which can also be tested using the **Manage Files** form (shown later on).

Form: **Add Booking**

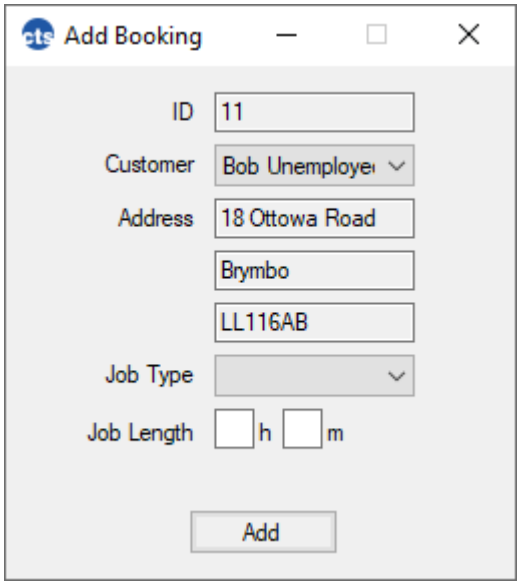
Test No.	Operation Tested	Event	Purpose	Expected Outcome	Observed Outcome
1	Allow user to select customer from customers.txt	Clicking on customer combobox	This will allow the user to select the desired customer from the existing list of customers. This makes selecting easier than typing the ID code, which the user will likely not know, and prevents the entering of invalid keys.	Dropdown box will contain the customers' names	**Dropdown box contained customer names**

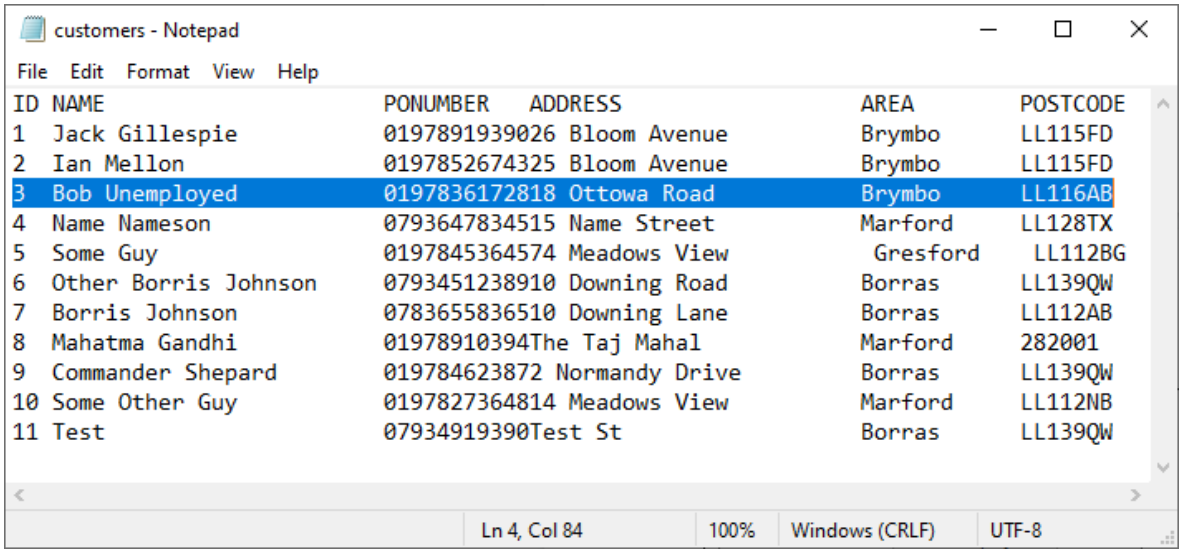




These screenshots show that the list is correctly composed of the names of the stored customers.

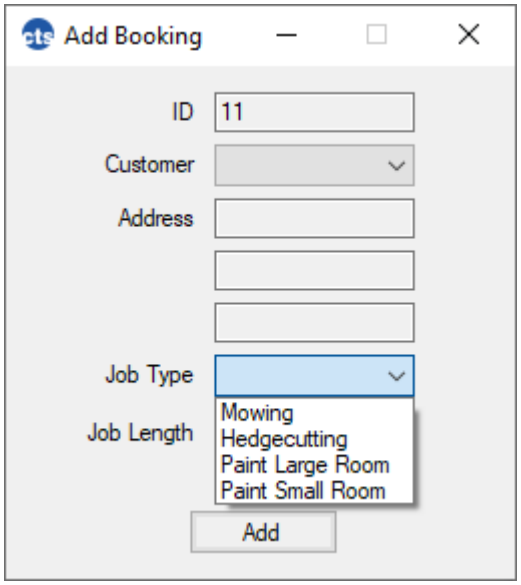
2	Fetch customer data once selected	Select an item from the customer drop down list.	This will autofill the address fields with the address of the selected customer. This helps to differentiate the customers, as if two customers share a name, the user can check if they have selected the correct one by checking the address. This reduces the likelihood of incorrect data handling, without having to make the users use ID keys.	Address fields will contain the customer's address	**Address fields contained the customer's address**

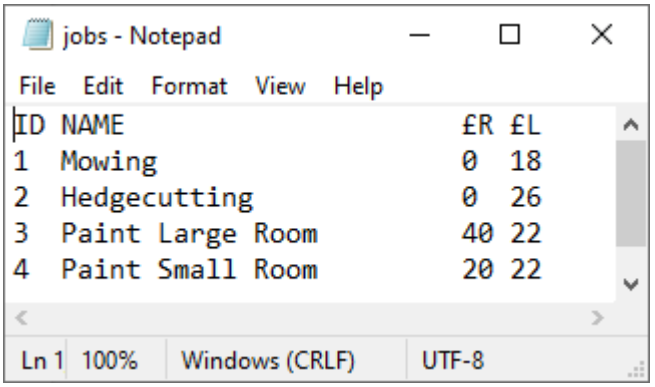




These screenshots show that the correct address has been fetched from the file and displayed in the boxes.

3	Allow user to select job from jobs.txt	Clicking on customer combobox	Similarly to test 1, this will allow the user to select jobs from the existing job list. This function is similar to the customer selection, however not the same piece of code, and so needs individual testing.	Dropdown box will contain the jobs' names	**Drop down box containe d the job names**

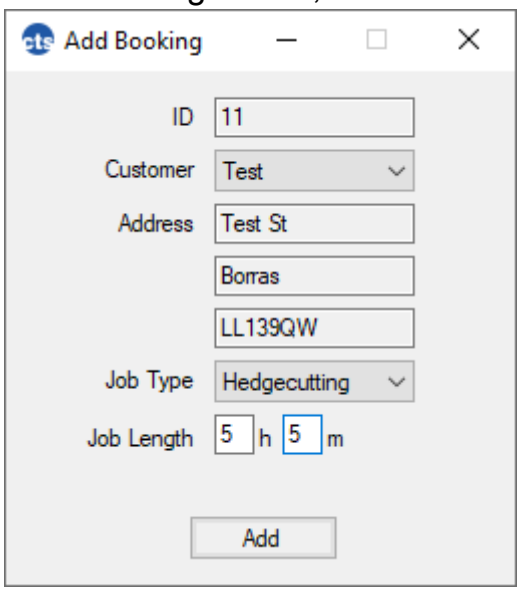


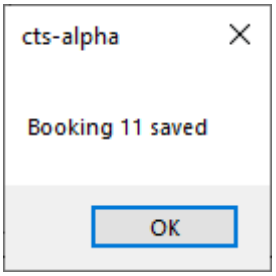


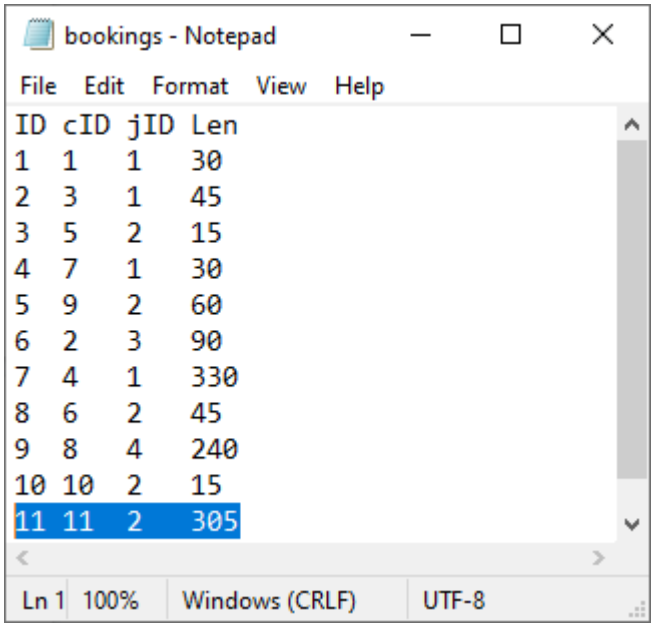
These screenshots show that the list is correctly composed of the jobs found in the jobs.txt file.

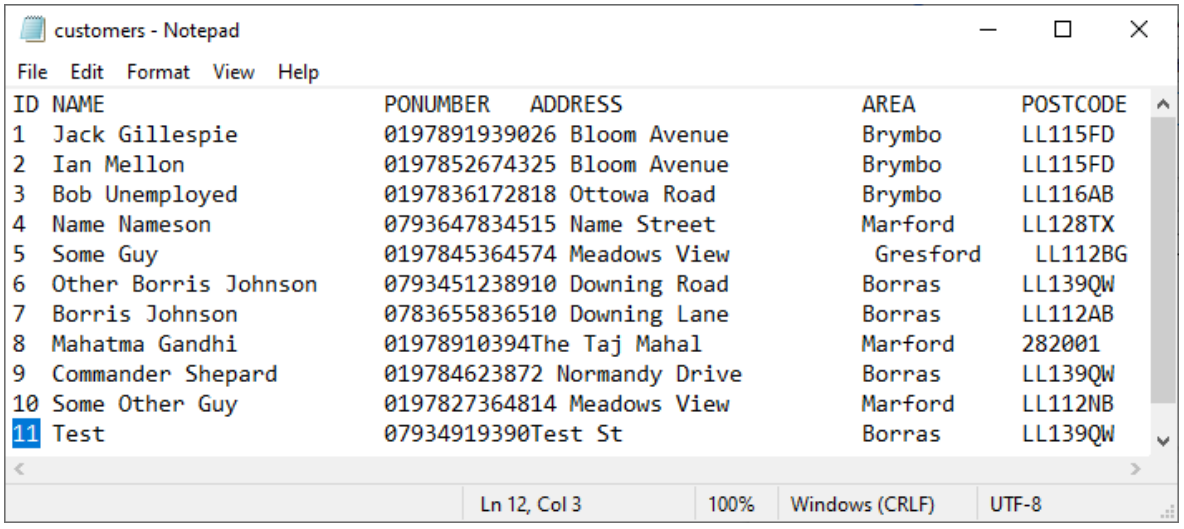
4	Store **correct** data	Pressing 'Add'	Ensures correct data is stored.	Data, consisting of customer ID and job ID is written to file.	

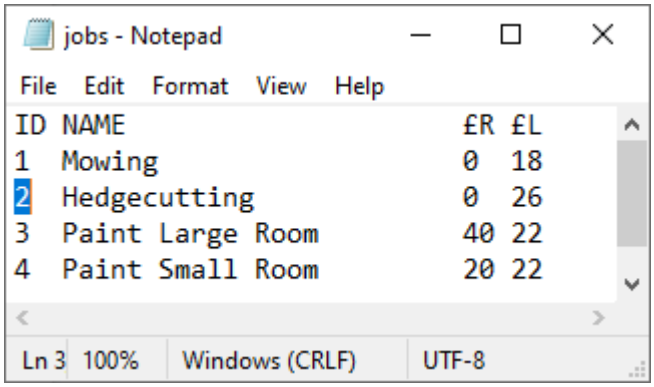
The ability to write to files using the fileHandling class has already been proven to work. However, this form selects the customer's and job's **names**, but then for the sake of normalisation, stores their **IDs**. This test ensures the IDs are being stored, and not the selected names.







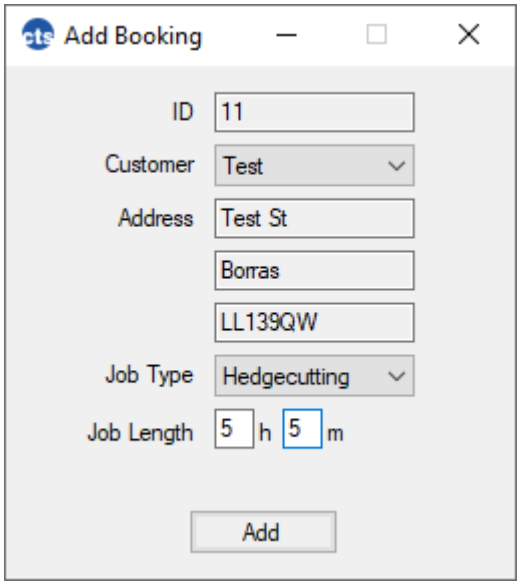


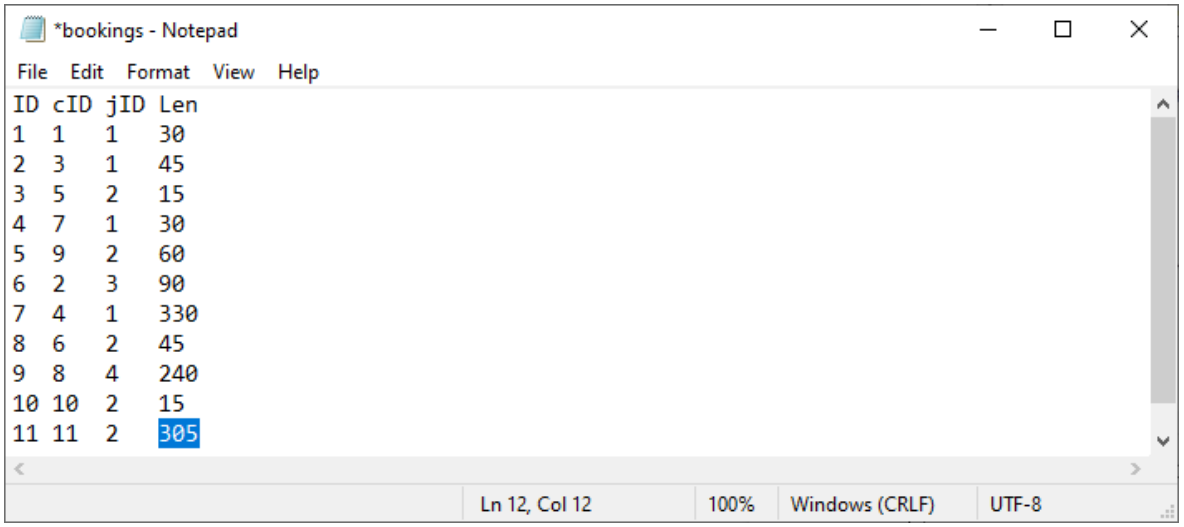


These screenshots show that the form has correctly stored the customer ID 11 (Test) and job ID 2 (Hedgecutting) into the bookings.txt file,which is correctly related to the customer and job files.

5	Time calculation	5h5m	The system stores time in terms of minutes,	5(60) + 5 = 305	**305**

			however the user inputs time in terms of hours and minutes, so this input needs to be converted to minutes to be stored.		

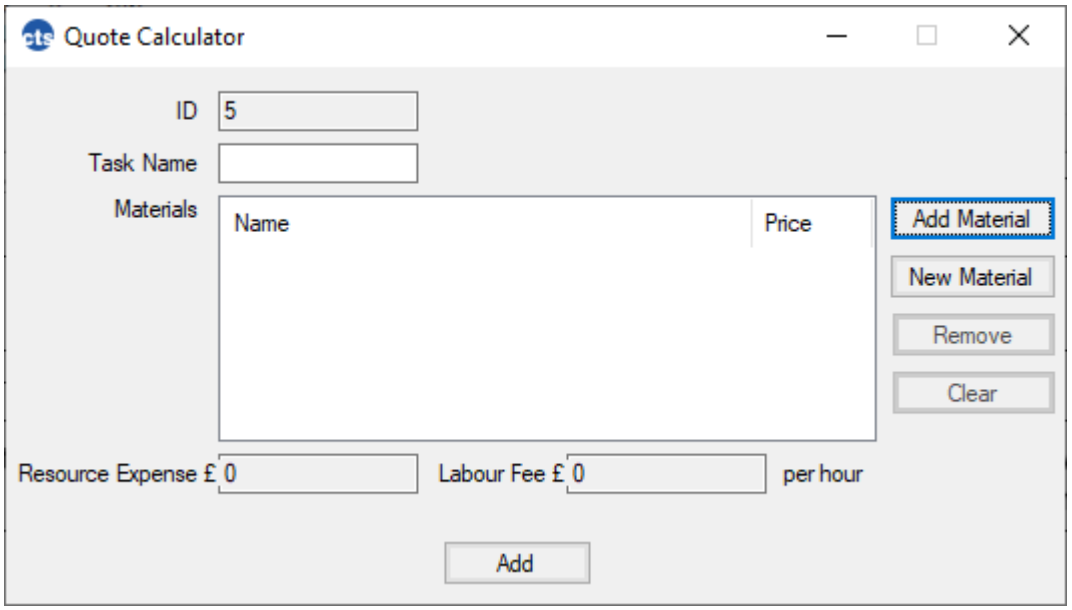


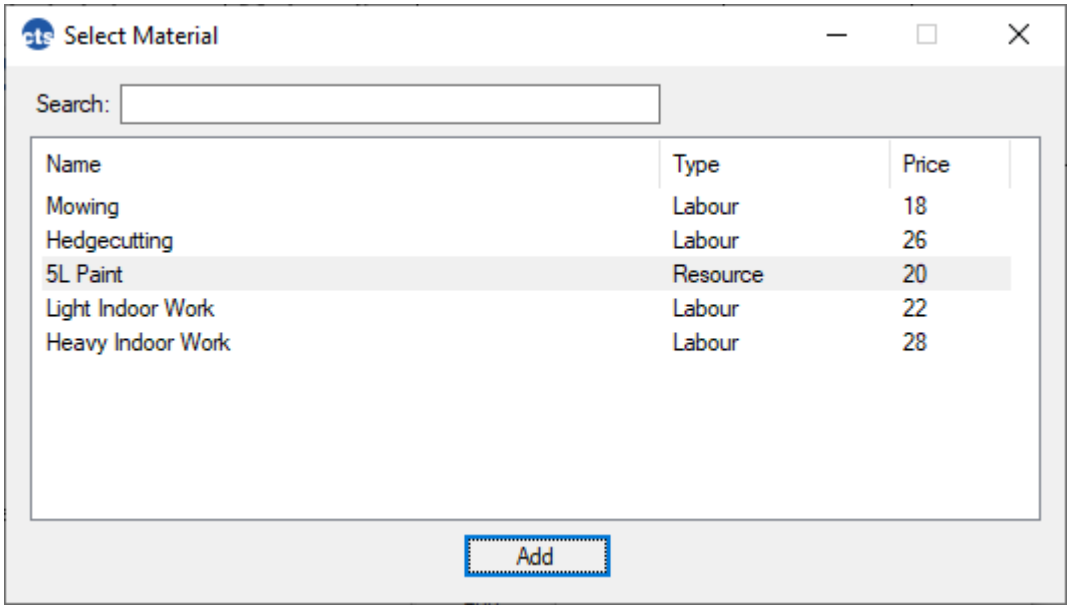


These screenshots show that the system has correctly converted 5h5m into 305 minutes, and stored that correct time in the file.

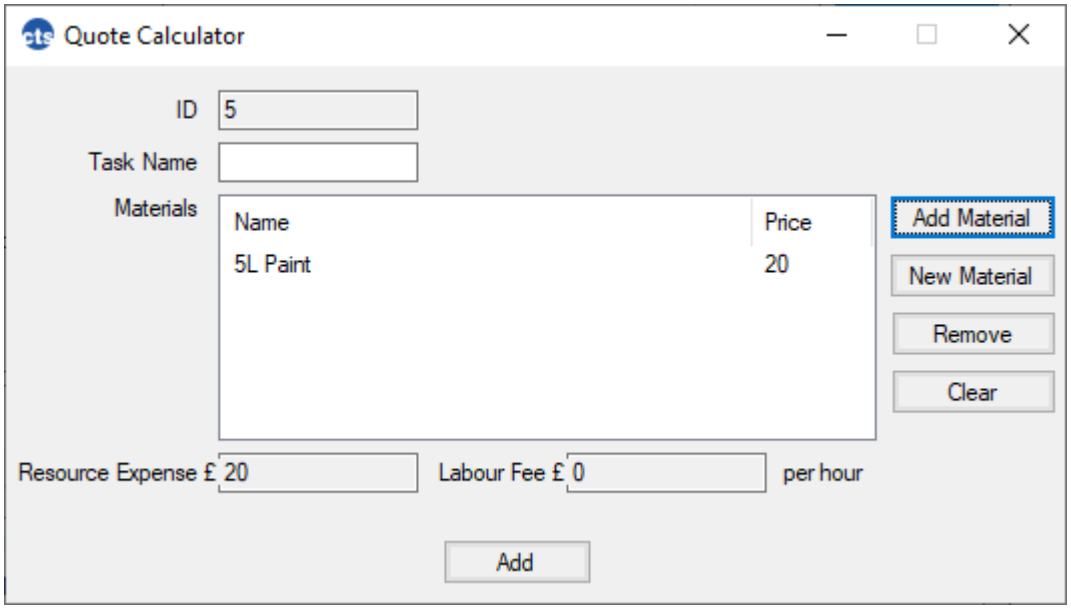
Form: **Create Quote** (using **Select Material**)					
Test No.	**Operation Tested**	**Event**	**Purpose**	**Expected Outcome**	**Observed Outcome**

1	Add material to list (using **Select Material** form)	Click ‘Add Material’, select material, click ‘Add’	This will allow the user to select the materials that are a part of this job.	5L paint in list. Resource/ labour prices should increase by the correct amount.	**5L paint added to the list. Resource Expense correctly increased.**



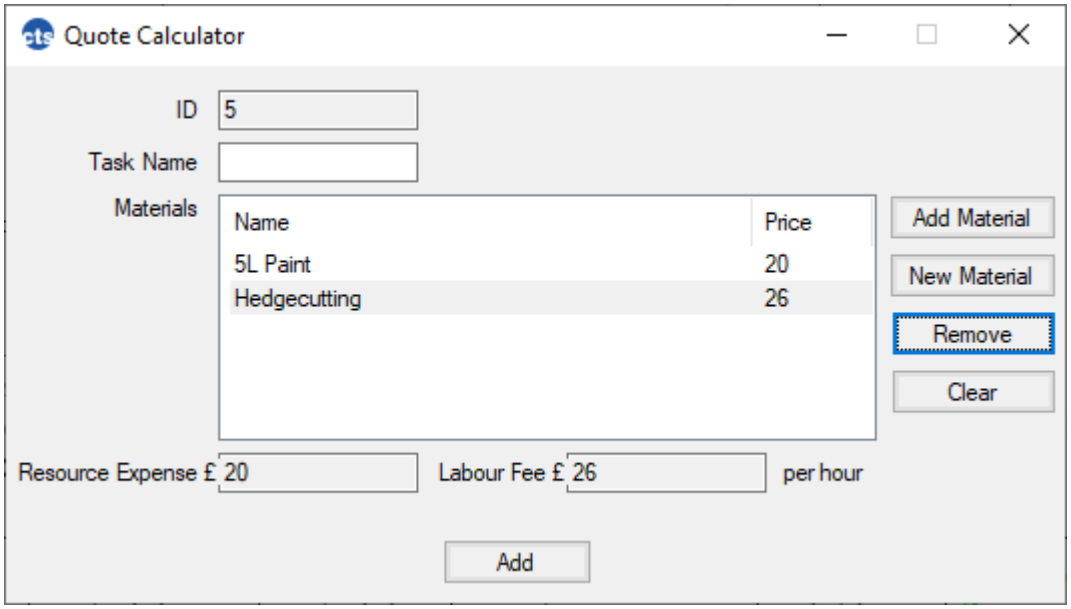


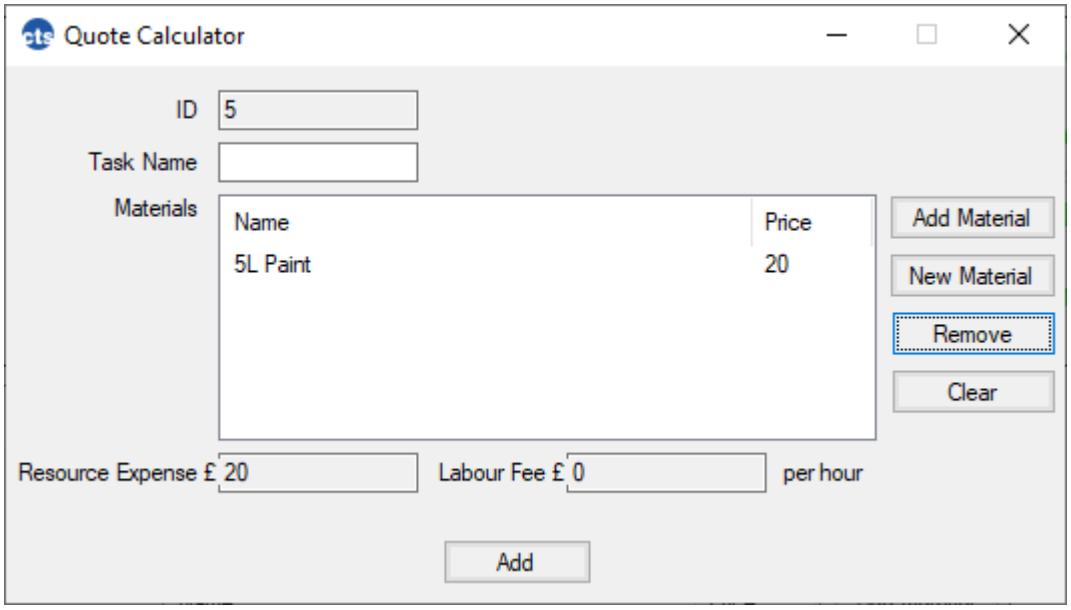
5L paint is a resource, worth £20.



These screenshots show that the form can successfully open the **Select Material** form, allowing the user to select a material, and add it to the list in **Quote Calculator** by pressing add. The item's correctly displayed in the list. And, the resource expense has correctly changed from 0 to 20.

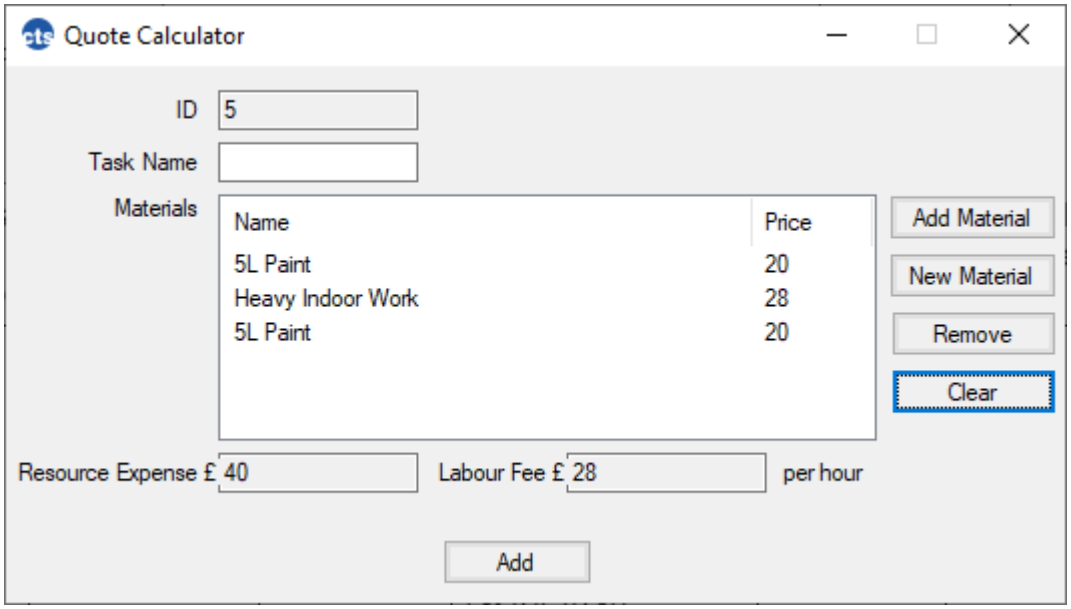
2	Remove material from list	Select material and press 'Remove Material'	This will allow the user to remove materials they have added, in the case they have added one that is not needed by accident.	Only 5L paint in the list.	**Hedgecutting removed from list, only 5L paint remains.**

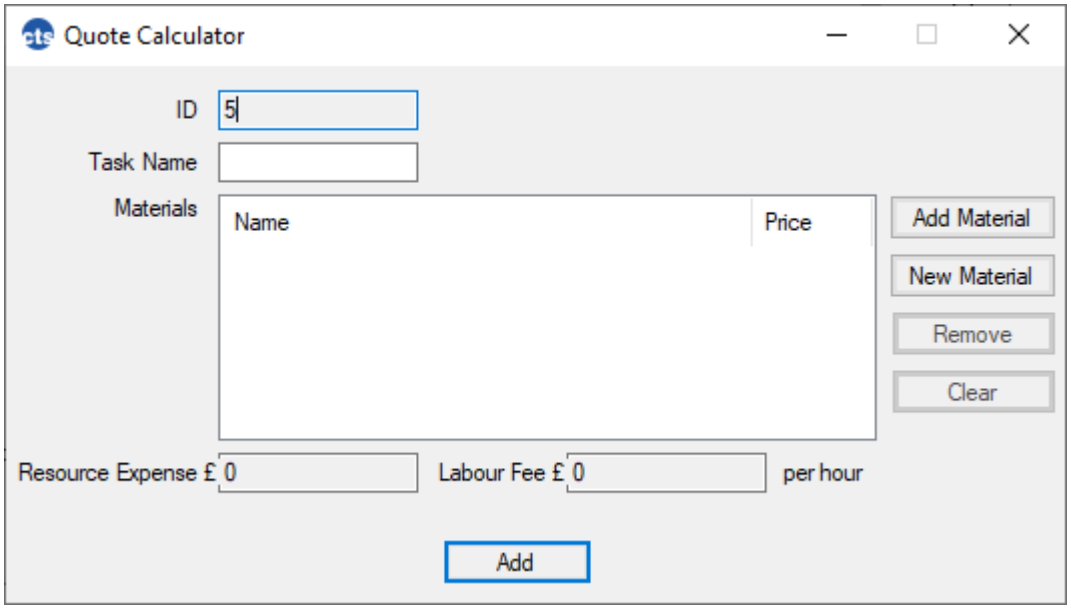




This screenshot shows that the items can be correctly removed from the list, and that the prices will also decrease respectively to ensure the prices are correct.

3	Clear list	Pressing ‘Clear’	This will allow the user to completely reset the list, in case all of the items need removing, as it is faster than removing them individually.	Empty list	**List is empty**





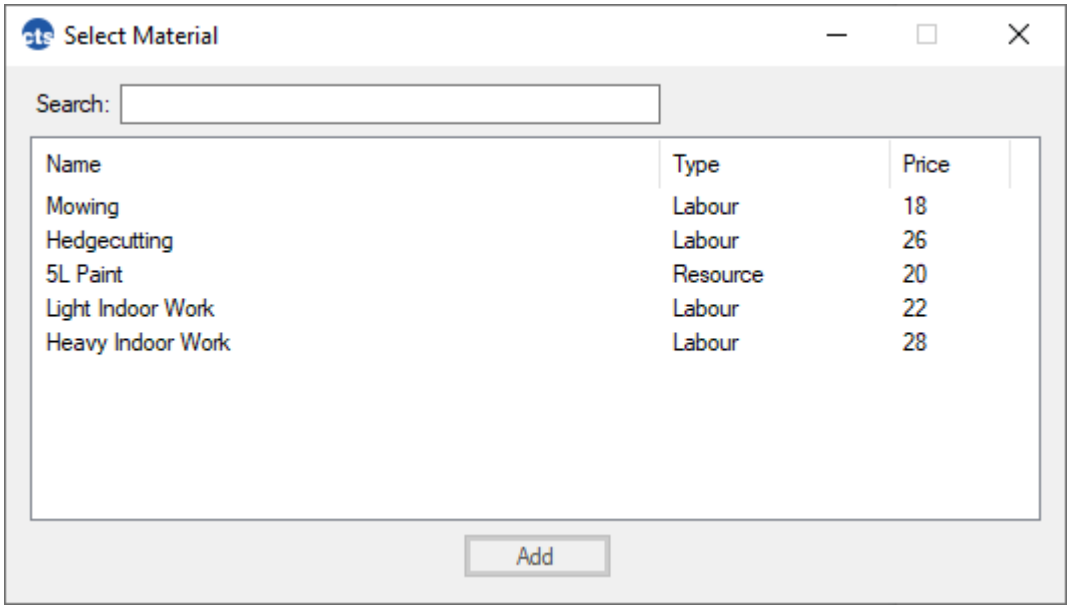
These screenshots show that the list has successfully been cleared, and that the prices have both reset to 0.

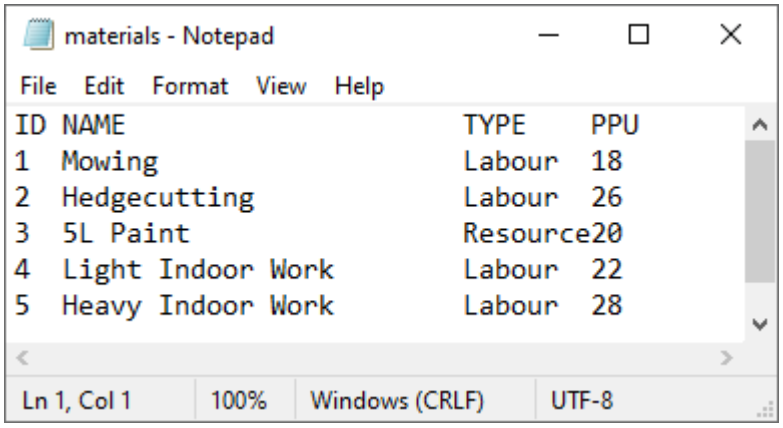
The ‘Add’ button stores the ID, Name and prices into the job.txt file using validation and the fileHandling class, both of which have been tested.

The ‘New Material’ button opens the **Add Material** form,which is the same form accessible from the main menu, which has been tested.

Form: **Select Material**

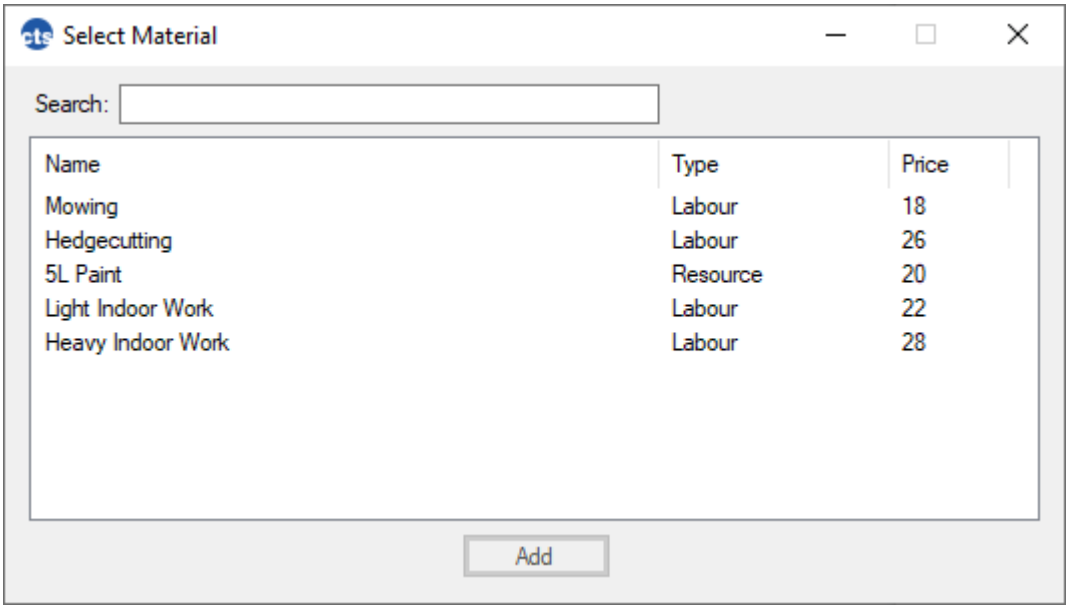
Test No.	**Operation Tested**	**Event**	**Purpose**	**Expected Outcome**	**Observed Outcome**
1	Load contents of materials.txt into list	On form load	This will allow the user to easily see and select the contents of the material.txt file, to ensure that the correct item is added to the **Create Quote** list.	Contents of jobs.txt correctly in list.	**Contents of jobs.txt correctly in list**

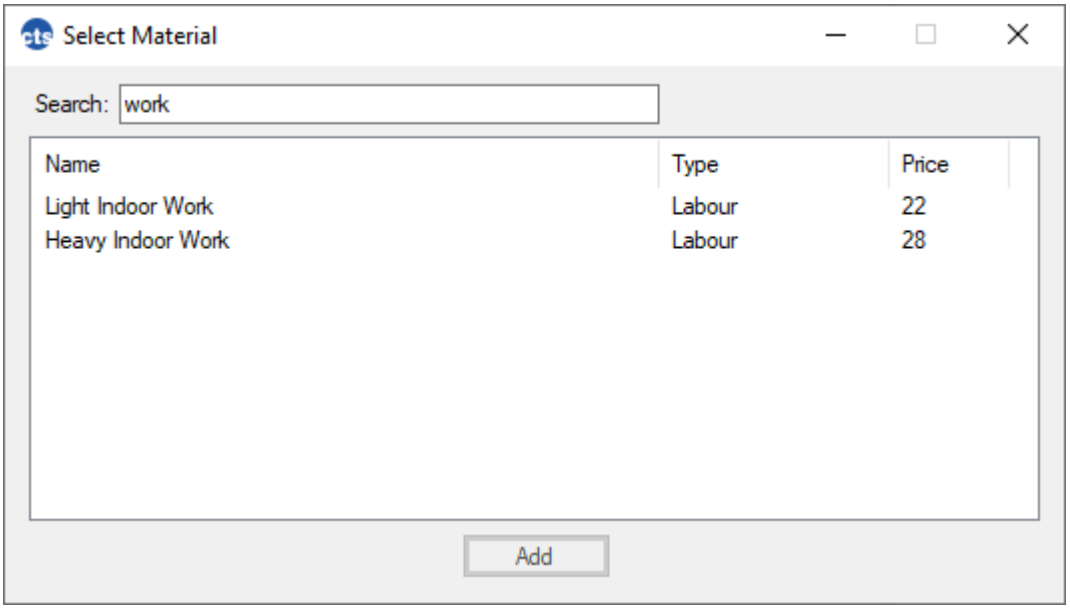




These Screenshots show that the form has correctly fetched the contents of materials.txt and is displaying it correctly in the table.

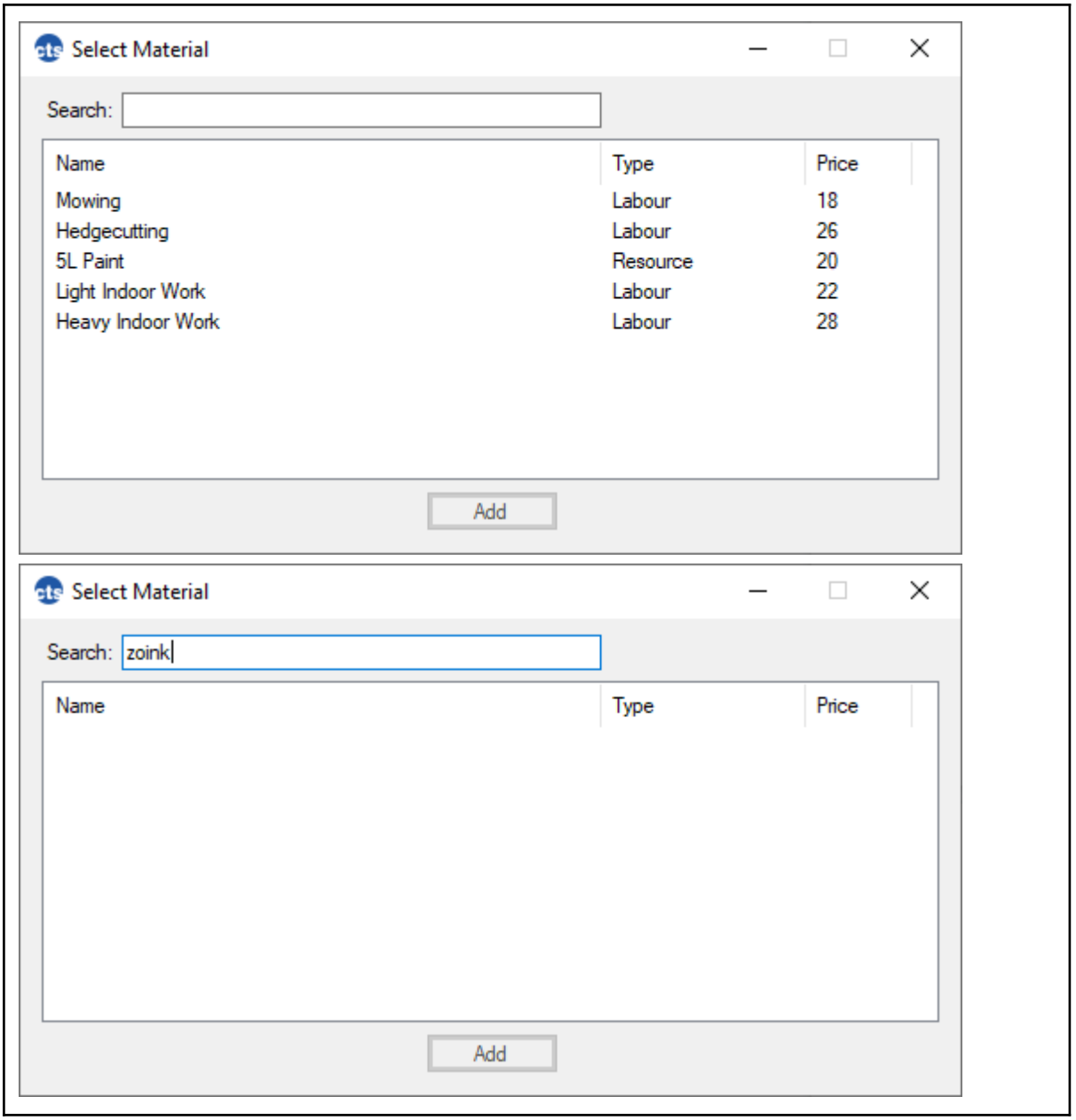
2	Search file contents	Input of data into search bar Valid data	This will allow the user to quickly find the item they want, instead of manually searching through the list.	Corresponding items should display	**Light Indoor Work and Heavy Indoor Work found.**





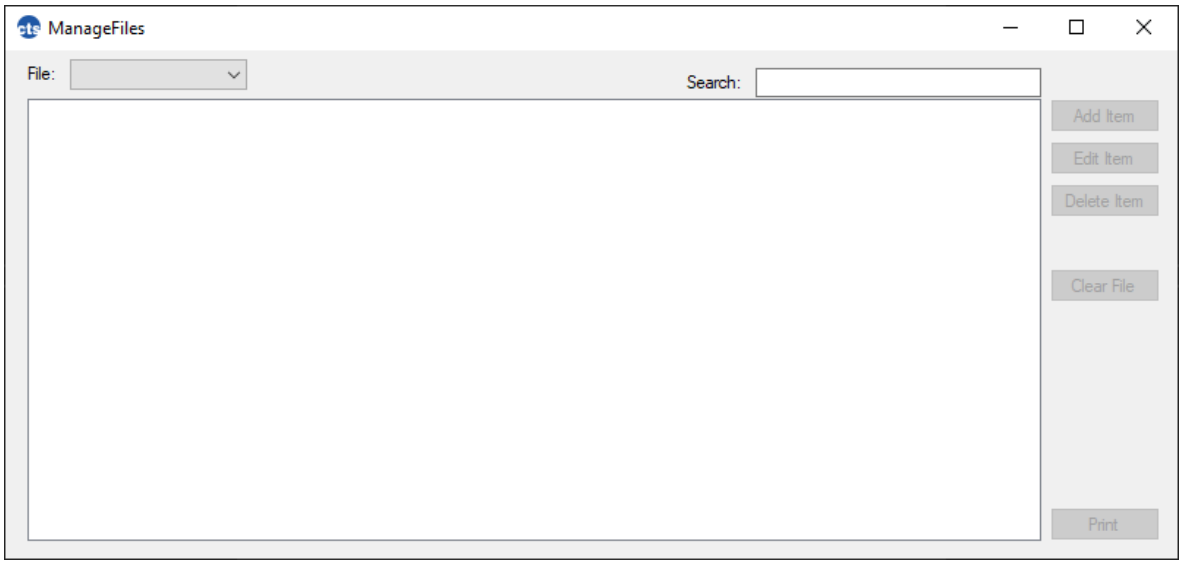
This screenshot shows that the correct items are still in the list, whereas items not containing ‘work’ are gone.
This also shows that search is **not** case sensitive, as both items contain the word ‘Work’, but are still found for the search ‘work’, which is more convenient for the user.

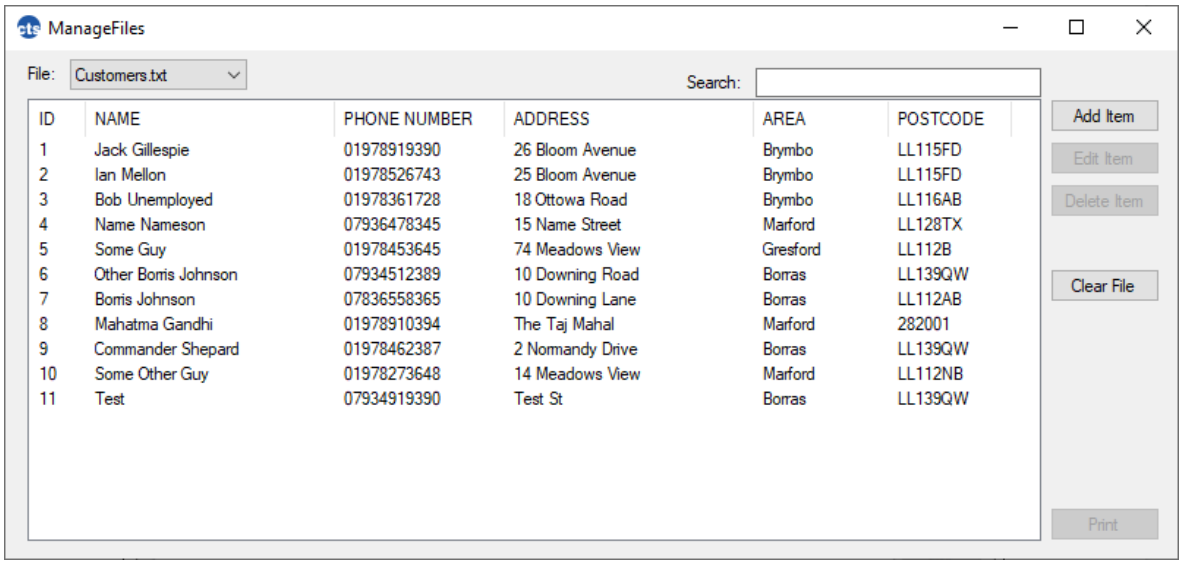
3	Search file contents	Input of data into search bar Invalid data	This will allow the user to quickly find the item they want, instead of manually searching through the list.	No items should be found.	**No items found.**

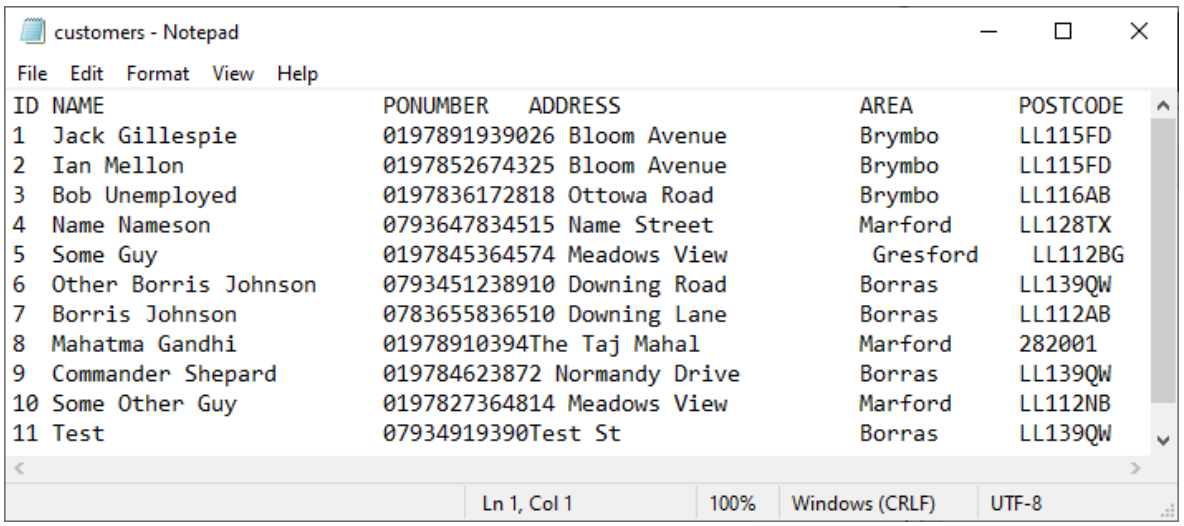


Form: **Manage Files**					
Test No.	**Operation Tested**	**Event**	**Purpose**	**Expected Outcome**	**Observed Outcome**
1	Load contents of selected file into list	Select file from list	This will allow the user to choose which file they wish to manage, and then display the data inthe list for easy and	Contents of customers .txt correctly in list.	**Contents of customers.txt correctly in list**

			convenient viewing.		

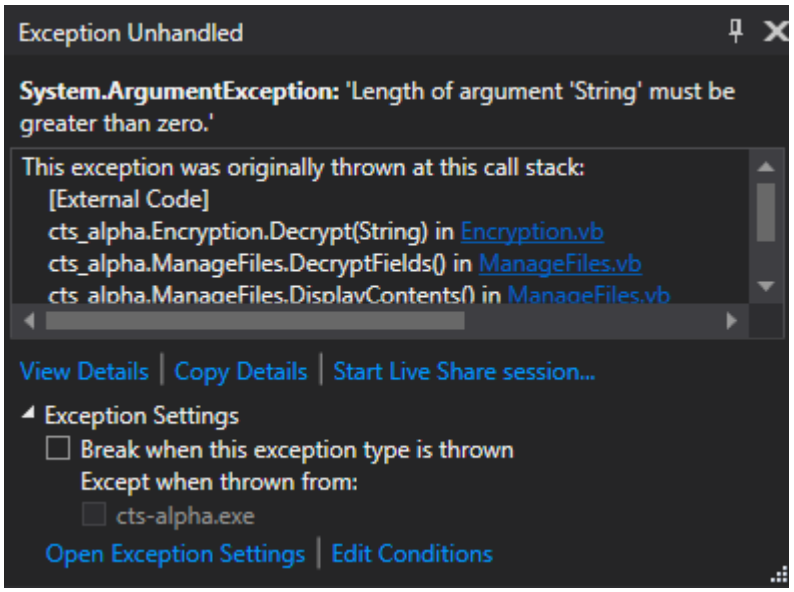






These screenshots show that the data from customers.txt has correctly been

fetched and displayed in the list. This process is the same for all of the files, with minor differences for **users.txt** and **bookings.txt** (see below).					
2	Decrypt contents of users.txt, to display.	Select users.txt	The contents of users.txt are encrypted, and so are meaningless without being decrypted.	Contents of users.txt correctly decrypted	**Runtime error**



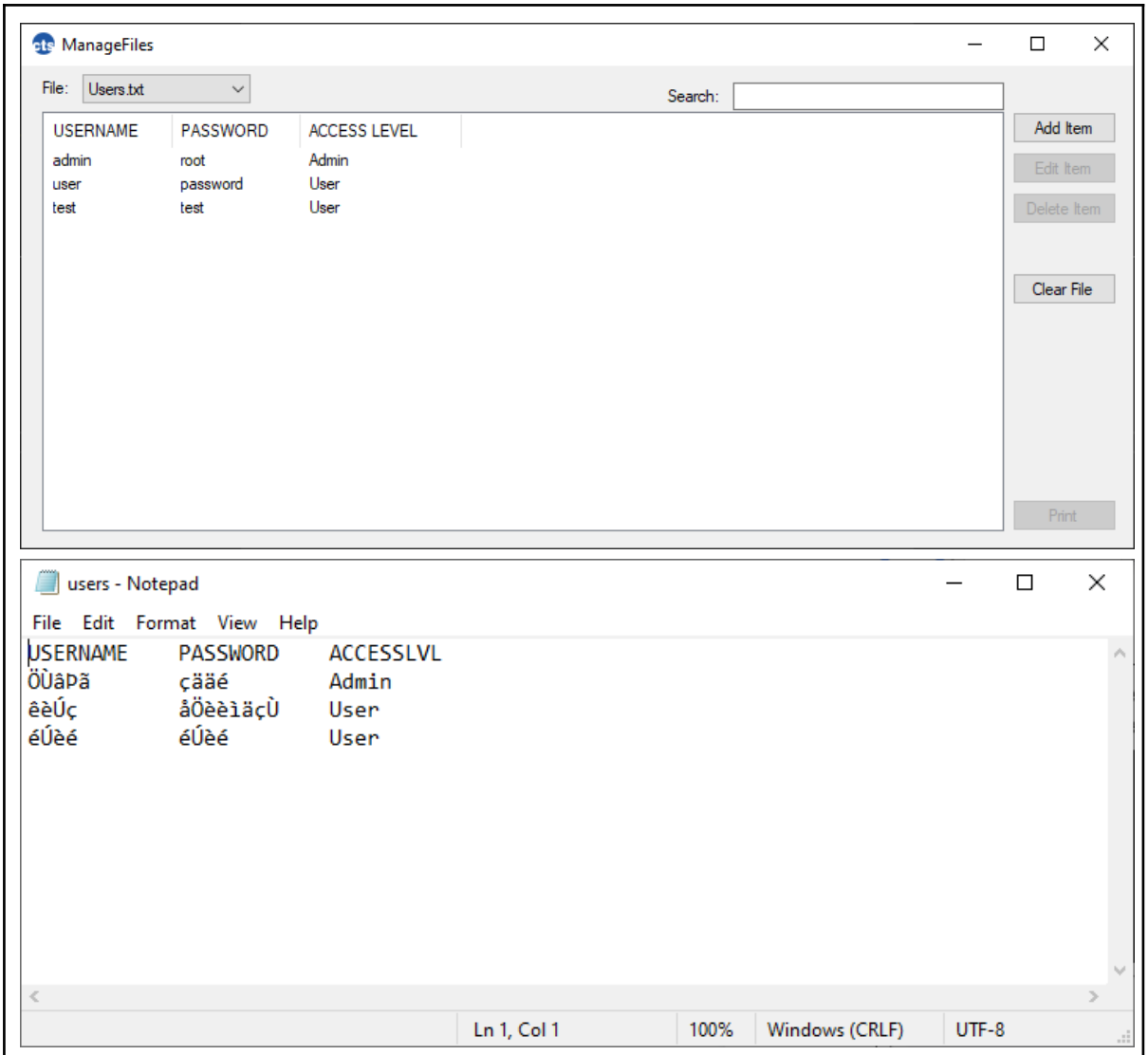
```
54      'OUTPUTS THE CONTENTS OF SELECTED FILE INTO LISTVIEW
        3 references
55      Public Sub DisplayContents()
56
57          If filenameBox.Text = "Users.txt" Then
58              DecryptFields()
59          End If
60
61          fileContents.Clear()
62          fileData = fileHandler.GetFileContents()
63
64          'COLUMNS (headers)
65          For x = 0 To dimensions.Length - 1
66              Dim newColumn = fileContents.Columns.Add(columns(x))
67
68              If dimensions(x) > columns(x).Length Then
69                  newColumn.Width = 20 + (6 * dimensions(x))
70              Else
71                  newColumn.Width = 20 + (8 * columns(x).Length)
72              End If
73
74          Next
75
76          'ROWS
77          For x = 1 To fileData.Count - 1
78
79              Dim sum = dimensions(0)
80              Dim newItem = fileContents.Items.Add(Trim(Mid(fileData(x), 1, sum))) 'first column
81
82              For n = 1 To dimensions.Length - 1
83
84                  If foreignKeys(n) = "" Then 'if this is not a foreign key, output data
85                      newItem.SubItems.Add(Trim(Mid(fileData(x), sum + 1, dimensions(n)))) 'nth column
86                      sum += dimensions(n)
87
88                  Else 'if this data is a foreign key, locate the data it refers to
89                      Dim foreignKey = Trim(Mid(fileData(x), sum + 1, 3)) 'key to locate
90
91                      'read primary data
92                      Dim tempFileHandler As New FileHandling(foreignKeys(n), header)
93                      Dim tempData() As String = tempFileHandler.GetFileContents()
94
95                      For y = 1 To tempData.Count - 1 'search for foreign key
96                          Dim tempID = Trim(Mid(tempData(y), 1, 3))
97                          If tempID = foreignKey Then
98                              newItem.SubItems.Add(Trim(Mid(tempData(y), 4, 25))) 'nth column
99                              sum += dimensions(n)
100
101                             fileContents.Columns(n).Width = 20 + (6 * 25)
102                         End If
103                     Next
104                 End If
105
106             Next
107
108         Next
109
110     End Sub
```

The error is caused as the program is attempting to decrypt the fields (line 58) **before** they are in the list.

2.1	Decrypt contents of users.txt, to display.	Select users.txt	The contents of users.txt are encrypted, and so are meaningless without being decrypted.	Contents of users.txt correctly decrypted	**Contents of users.txt correctly decrypted**

```
54      'OUTPUTS THE CONTENTS OF SELECTED FILE INTO LISTVIEW
        3 references
55      Public Sub DisplayContents()
56
57          fileContents.Clear()
58          fileData = fileHandler.GetFileContents()
59
60          'COLUMNS (headers)
61          For x = 0 To dimensions.Length - 1
62              Dim newColumn = fileContents.Columns.Add(columns(x))
63
64              If dimensions(x) > columns(x).Length Then
65                  newColumn.Width = 20 + (6 * dimensions(x))
66              Else
67                  newColumn.Width = 20 + (8 * columns(x).Length)
68              End If
69
70          Next
71
72          'ROWS
73          For x = 1 To fileData.Count - 1
74
75              Dim sum = dimensions(0)
76              Dim newItem = fileContents.Items.Add(Trim(Mid(fileData(x), 1, sum))) 'first column
77
78              For n = 1 To dimensions.Length - 1
79
80                  If foreignKeys(n) = "" Then 'if this is not a foreign key, output data
81                      newItem.SubItems.Add(Trim(Mid(fileData(x), sum + 1, dimensions(n)))) 'nth column
82                      sum += dimensions(n)
83
84                  Else 'if this data is a foreign key, locate the data it refers to
85                      Dim foreignKey = Trim(Mid(fileData(x), sum + 1, 3)) 'key to locate
86
87                      'read primary data
88                      Dim tempFileHandler As New FileHandling(foreignKeys(n), header)
89                      Dim tempData() As String = tempFileHandler.GetFileContents()
90
91                      For y = 1 To tempData.Count - 1 'search for foreign key
92                          Dim tempID = Trim(Mid(tempData(y), 1, 3))
93                          If tempID = foreignKey Then
94                              newItem.SubItems.Add(Trim(Mid(tempData(y), 4, 25))) 'nth column
95                              sum += dimensions(n)
96
97                              fileContents.Columns(n).Width = 20 + (6 * 25)
98                          End If
99                      Next
100                 End If
101
102             Next
103
104         Next
105
106         If filenameBox.Text = "Users.txt" Then
107             DecryptFields()
108         End If
109
110     End Sub
```

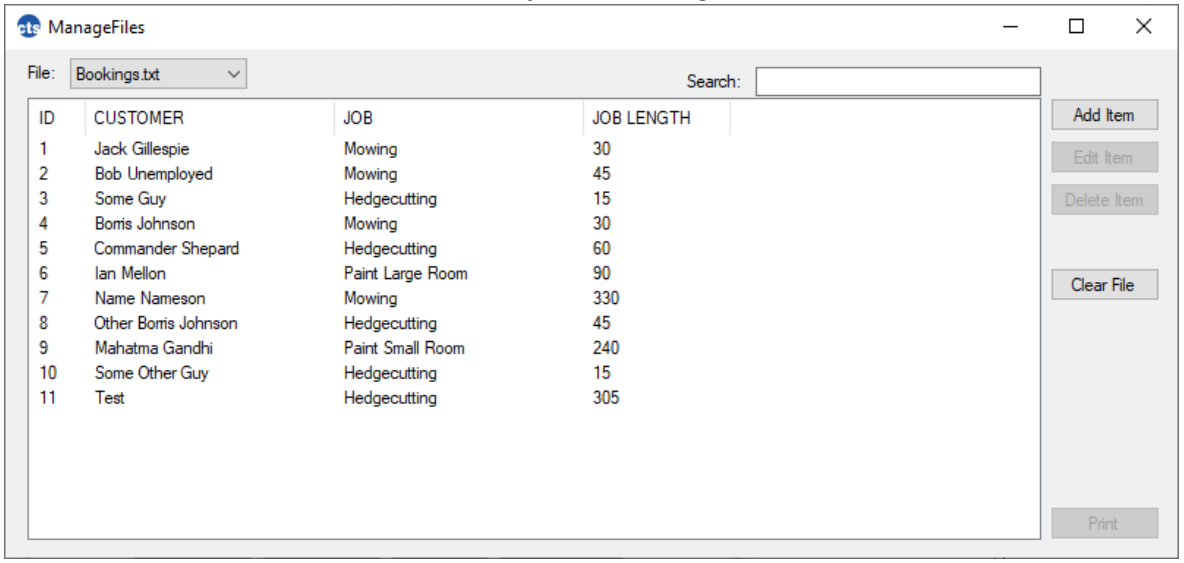
This amended code decrypts the contents **after** the list has its contents (line 107).

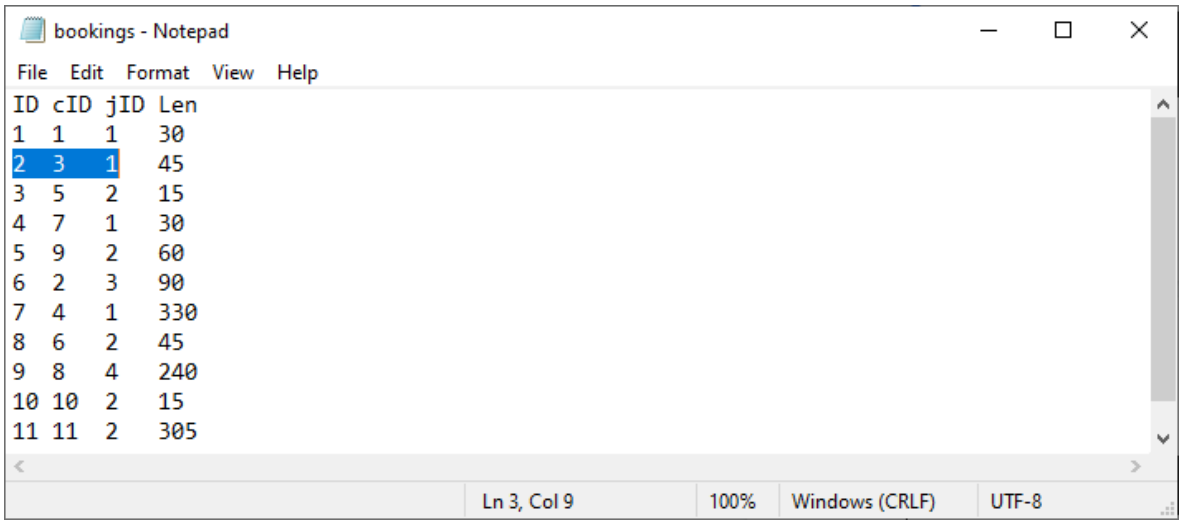


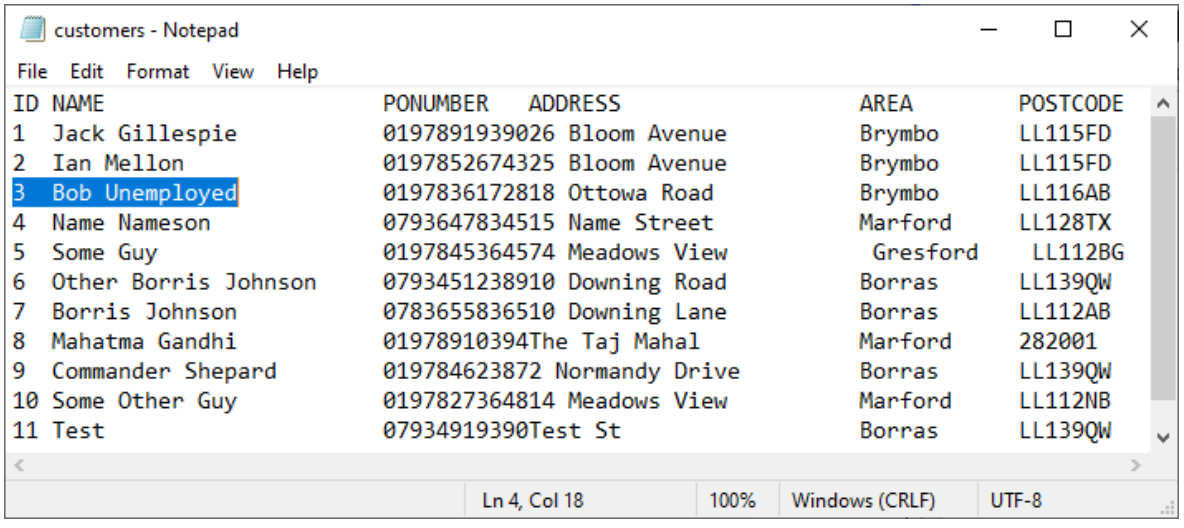
3	Find the name of items using foreign and primary keys	Select bookings.txt	This will display the name of the customers and jobs found in the bookings, instead of displaying their ID key, which is meaningless to the user.	Correct contentsof bookings.txt displayed, with cID and jID fields replaced by their respective names found in the customers .txt and jobs.txt	**Correct data.**

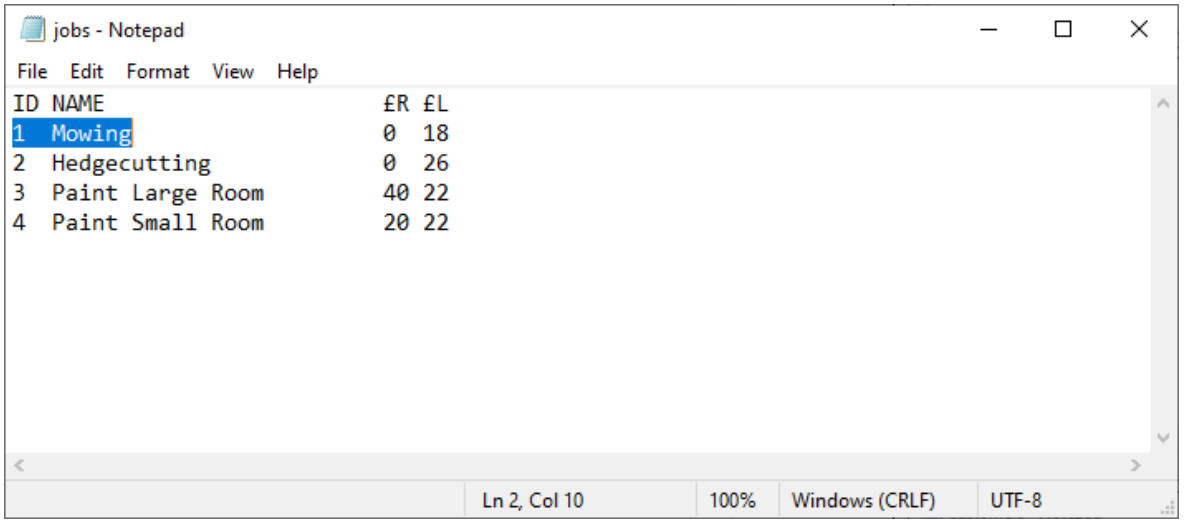
				files.	

For this test we will look specifically at booking 2.



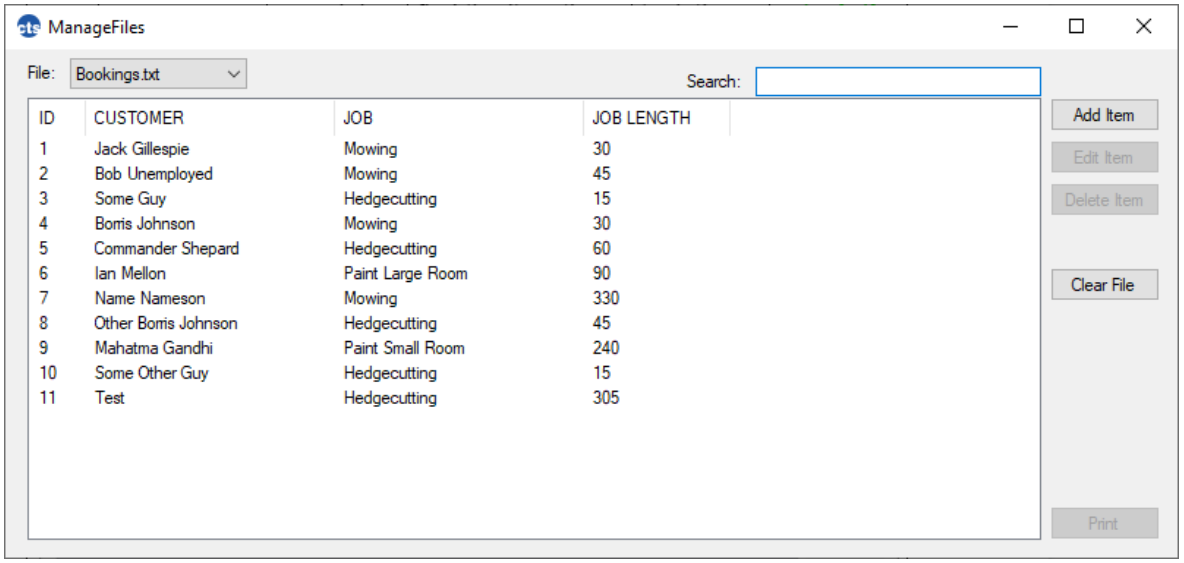


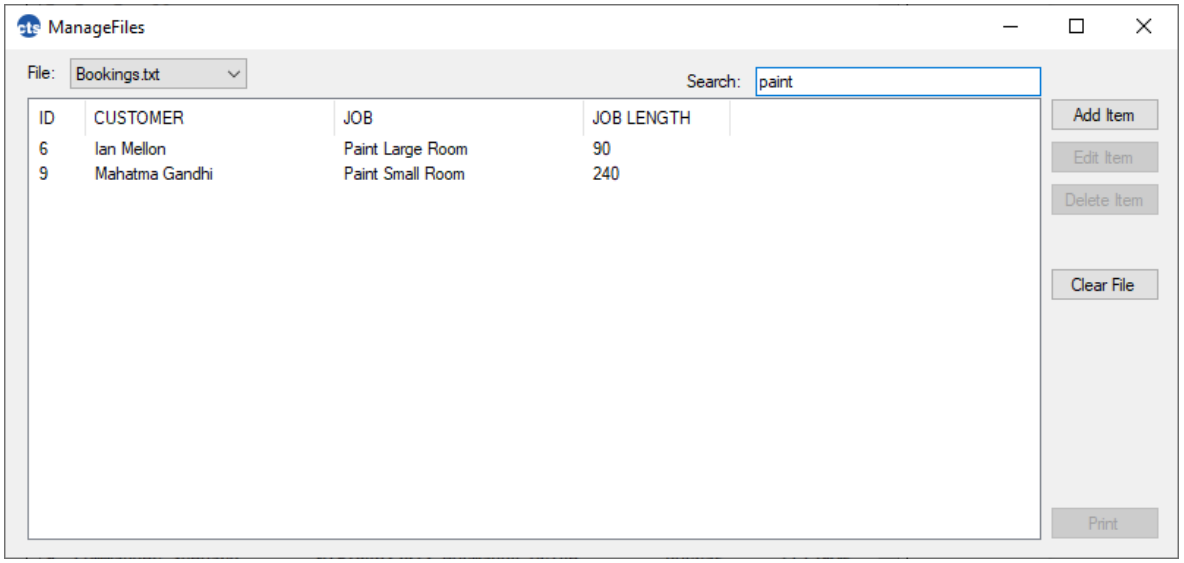




The first screenshot shows booking 2 as Bob Unemployed and Mowing. The bookings.txt shows that the stored data is cID 3 (Bob Unemployed) and jID 1 (Mowing), meaning it is displaying the correct data.

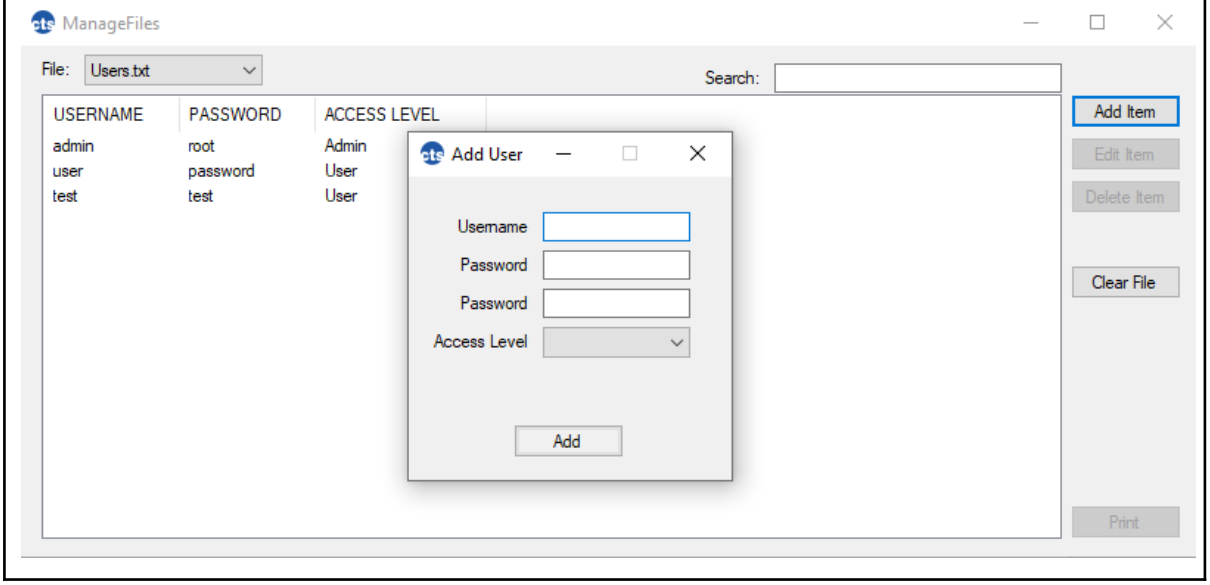
4	Use the search bar	Enter ‘paint’ into search bar	This will allow the user to quickly find the item they want, instead of manually searching through the list.	Items containing ‘paint’	**Items containing ‘paint’**

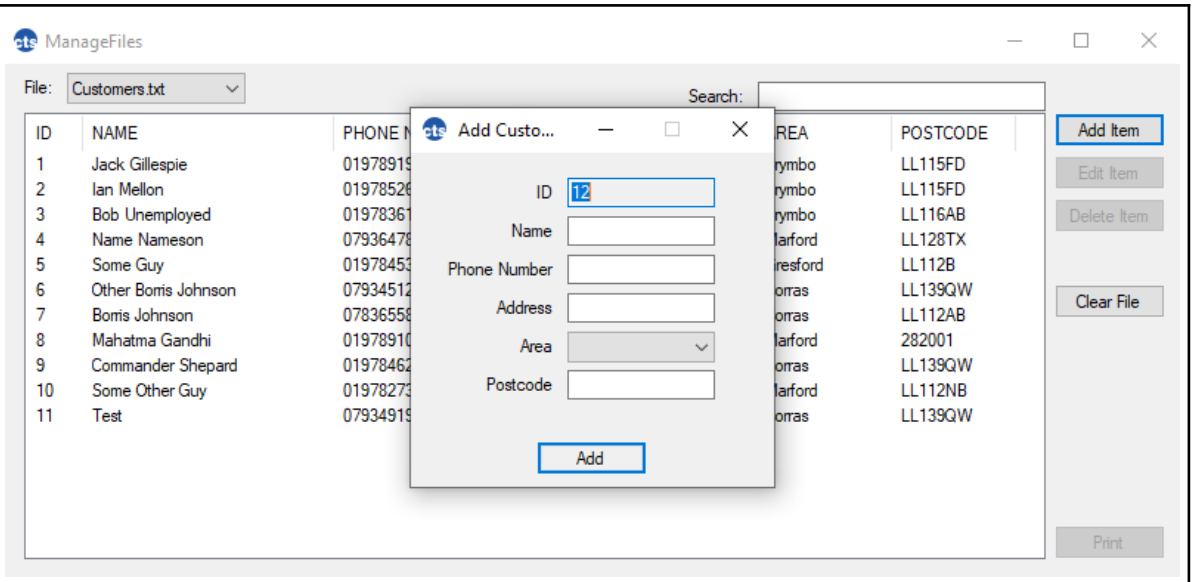




This screenshot shows that the search function only outputs items containing the word ‘paint’, in this case items 6 and 9.
This also shows that search is **not** case sensitive, as both items contain the word ‘Paint’, but are still found for the search ‘paint’, which is more convenient for the user.

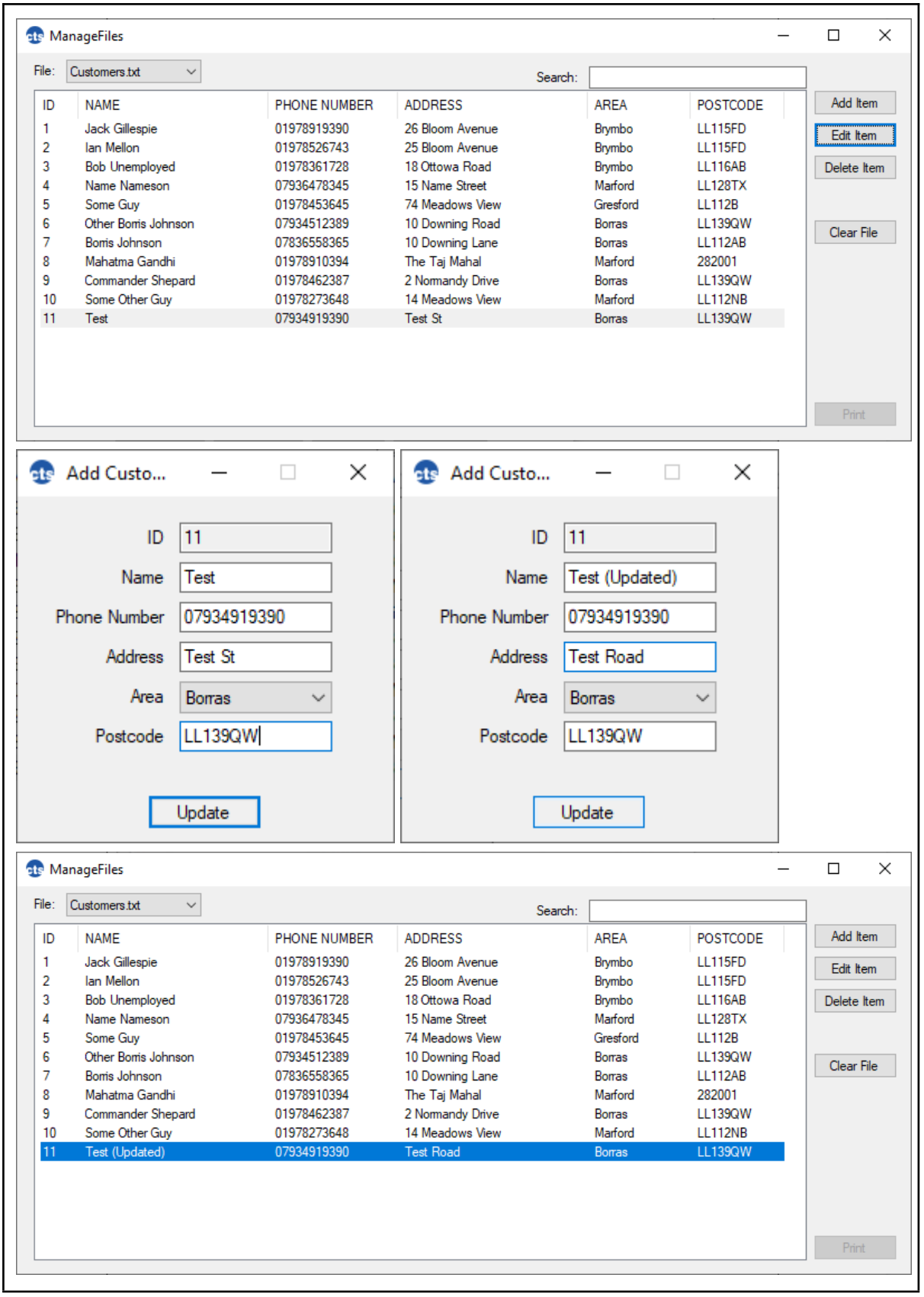
5	Add a new entry to the selected file.	Use the ‘Add Item’ button	This will open the **Add** form for the currently selected file, allowing the user to add more items to thefile.	Correct form opened	**Add User opened**

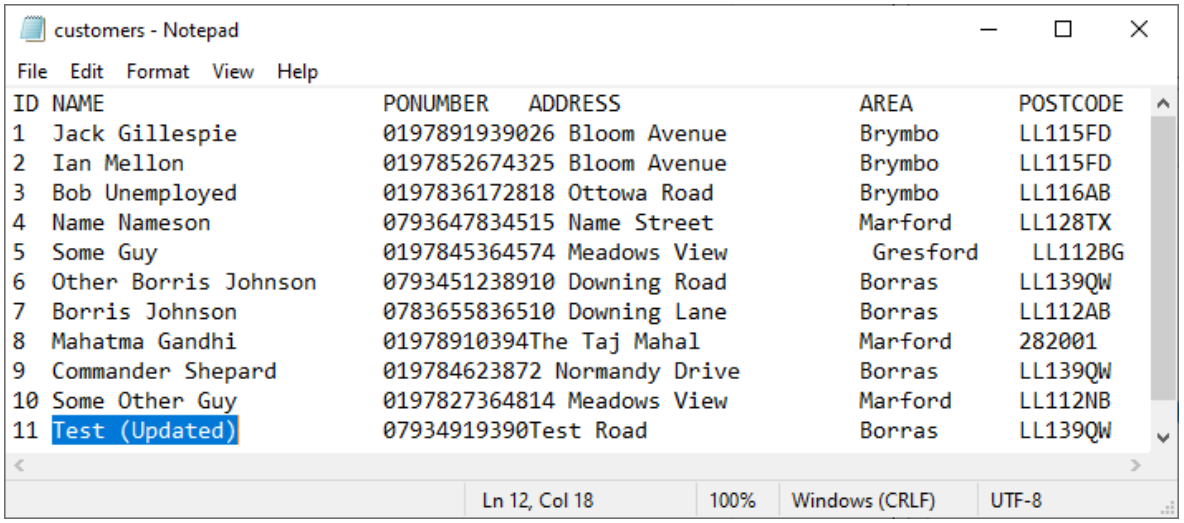




This screenshot shows that the ‘Add Item’ button opens the correct add form. These forms are the same ones accessible from the main menu, and so are already fully tested.

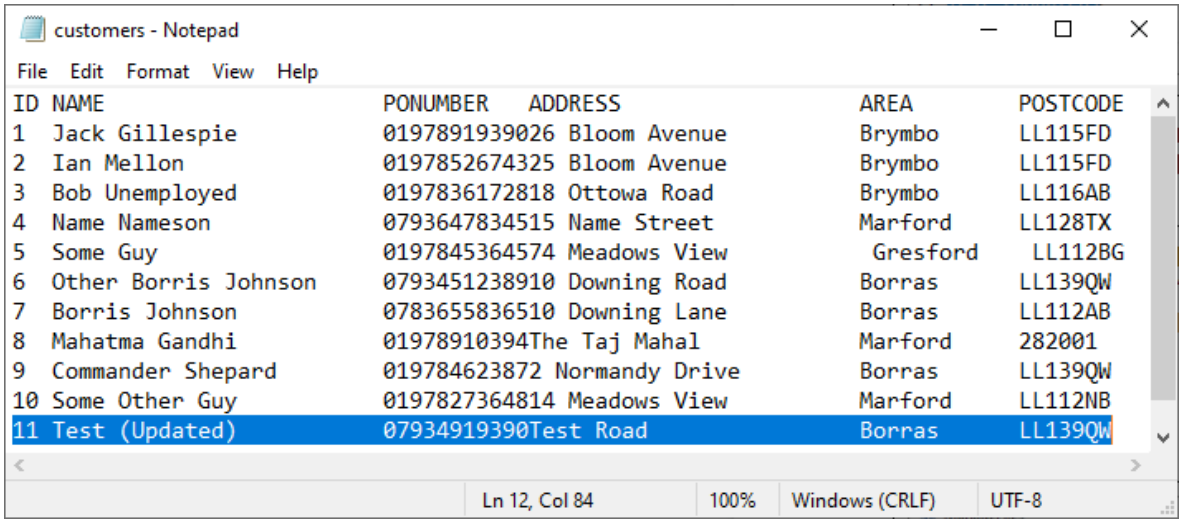
6	Edit an entry in the selected file.	Use the ‘Edit Item’ button	This will open the **Add** form for the currently selected file, allowing the user to add more items to the file. This form will be the same as the ‘Add Item’ form. However, the text will be altered to say ‘Update’ and not add, and will use the fileHandler slightly differently.	Correct data edited correctly.	**Correct data edited correctly**

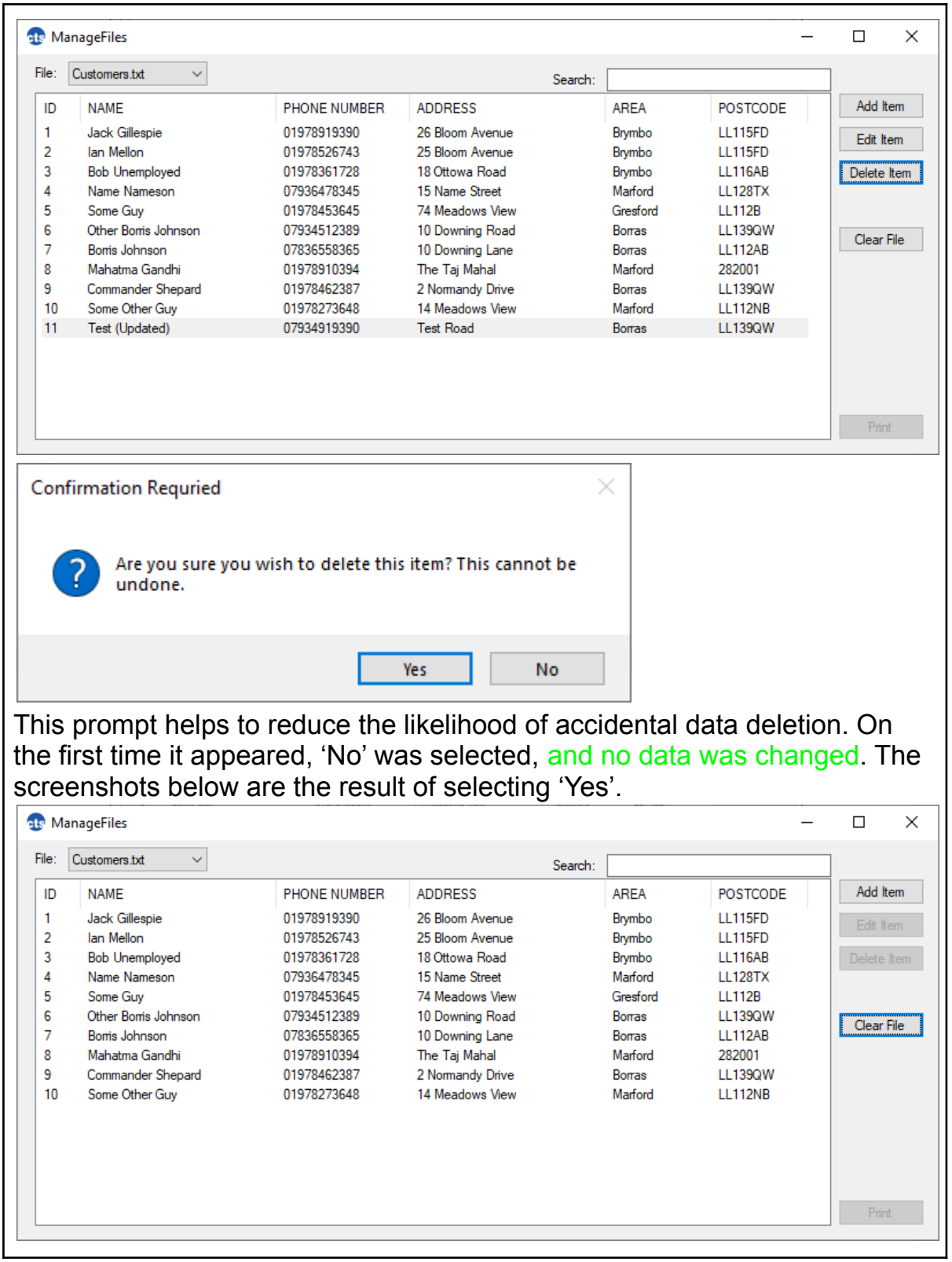

ManageFiles
File: Customers.txt
Search:
ID NAME PHONE NUMBER ADDRESS AREA POSTCODE
1 Jack Gillespie 01978919390 26 Bloom Avenue Brymbo LL115FD
2 Ian Mellon 01978526743 25 Bloom Avenue Brymbo LL115FD
3 Bob Unemployed 01978361728 18 Ottowa Road Brymbo LL116AB
4 Name Nameson 07936478345 15 Name Street Marford LL128TX
5 Some Guy 01978453645 74 Meadows View Gresford LL112B
6 Other Borris Johnson 07934512389 10 Downing Road Borras LL139QW
7 Borris Johnson 07836558365 10 Downing Lane Borras LL112AB
8 Mahatma Gandhi 01978910394 The Taj Mahal Marford 282001
9 Commander Shepard 01978462387 2 Normandy Drive Borras LL139QW
10 Some Other Guy 01978273648 14 Meadows View Marford LL112NB
11 Test 07934919390 Test St Borras LL139QW
Add Item
Edit Item
Delete Item
Clear File
Print
Add Custo...
ID 11
Name Test
Phone Number 07934919390
Address Test St
Area Borras
Postcode LL139QW
Update
Add Custo...
ID 11
Name Test (Updated)
Phone Number 07934919390
Address Test Road
Area Borras
Postcode LL139QW
Update
ManageFiles
File: Customers.txt
Search:
ID NAME PHONE NUMBER ADDRESS AREA POSTCODE
1 Jack Gillespie 01978919390 26 Bloom Avenue Brymbo LL115FD
2 Ian Mellon 01978526743 25 Bloom Avenue Brymbo LL115FD
3 Bob Unemployed 01978361728 18 Ottowa Road Brymbo LL116AB
4 Name Nameson 07936478345 15 Name Street Marford LL128TX
5 Some Guy 01978453645 74 Meadows View Gresford LL112B
6 Other Borris Johnson 07934512389 10 Downing Road Borras LL139QW
7 Borris Johnson 07836558365 10 Downing Lane Borras LL112AB
8 Mahatma Gandhi 01978910394 The Taj Mahal Marford 282001
9 Commander Shepard 01978462387 2 Normandy Drive Borras LL139QW
10 Some Other Guy 01978273648 14 Meadows View Marford LL112NB
11 Test (Updated) 07934919390 Test Road Borras LL139QW
Add Item
Edit Item
Delete Item
Clear File
Print



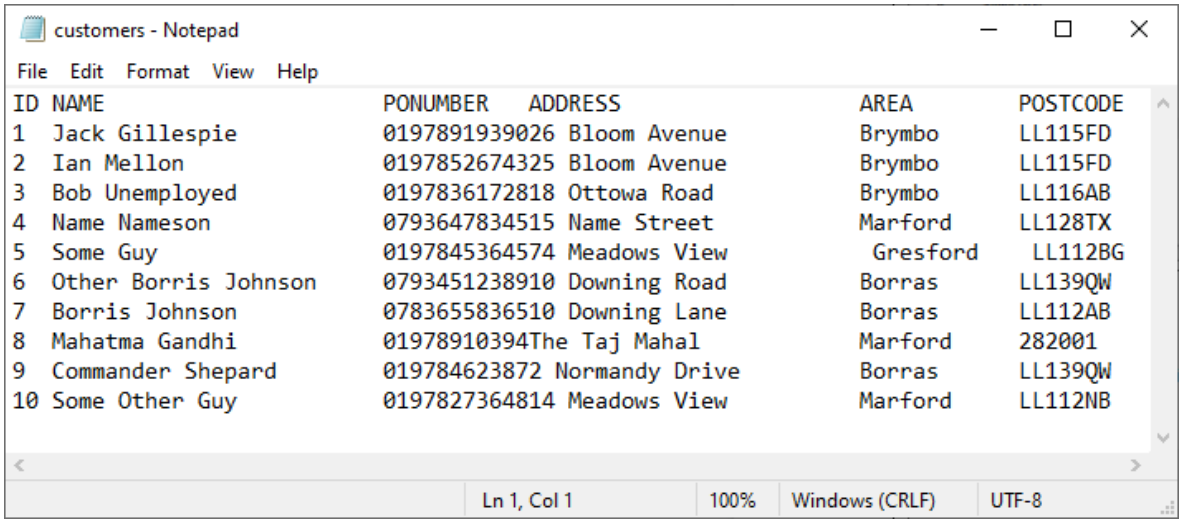
These screenshots show that the system correctly opens the form, with the original data inside, and allows the user to change the data, and then update the record in the file (as shown in the latter screenshot, where the data has changed).

7	Delete item from file	Press ‘Delete Item’	This will allow the user to remove an item in that case that it is no longer needed or wanted in the system.	Selected item removed from file.	**Selected item removed from file**



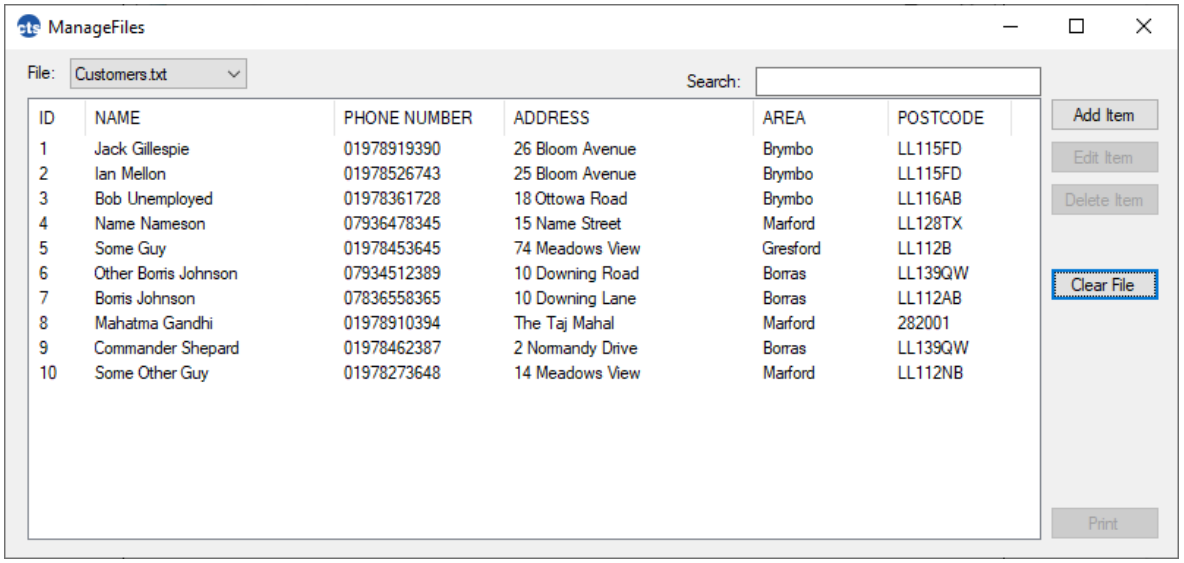


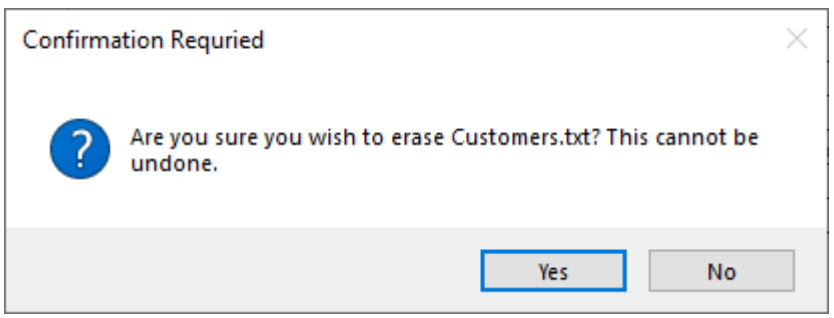
This prompt helps to reduce the likelihood of accidental data deletion. On the first time it appeared, 'No' was selected, and no data was changed. The screenshots below are the result of selecting 'Yes'.



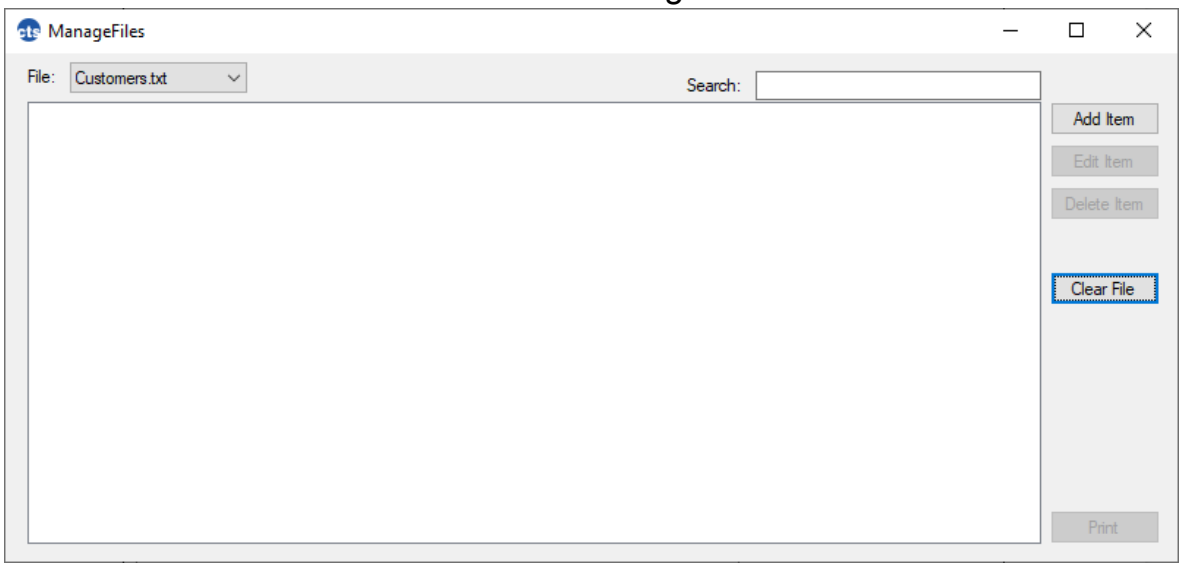
These screenshots show that the selected item has been deleted from the customers.txt file, and is also no longer present in the **Manage Files** list either.

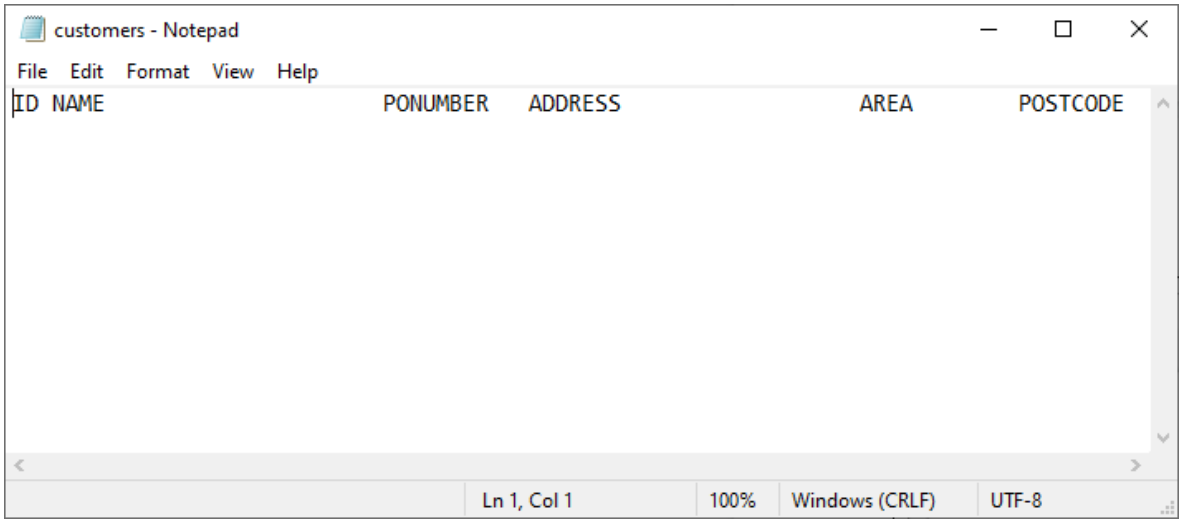
8	Wipe a file	Press ‘Clear’	This will allow the user to remove all data items from a file. This is good if a reset is required.	All items removed from file.	**All items removed from file. Header intact.**





This prompt helps to reduce the likelihood of accidental data deletion. On the first time it appeared, ‘No’ was selected, and no data was changed. The screenshots below are the result of selecting ‘Yes’.

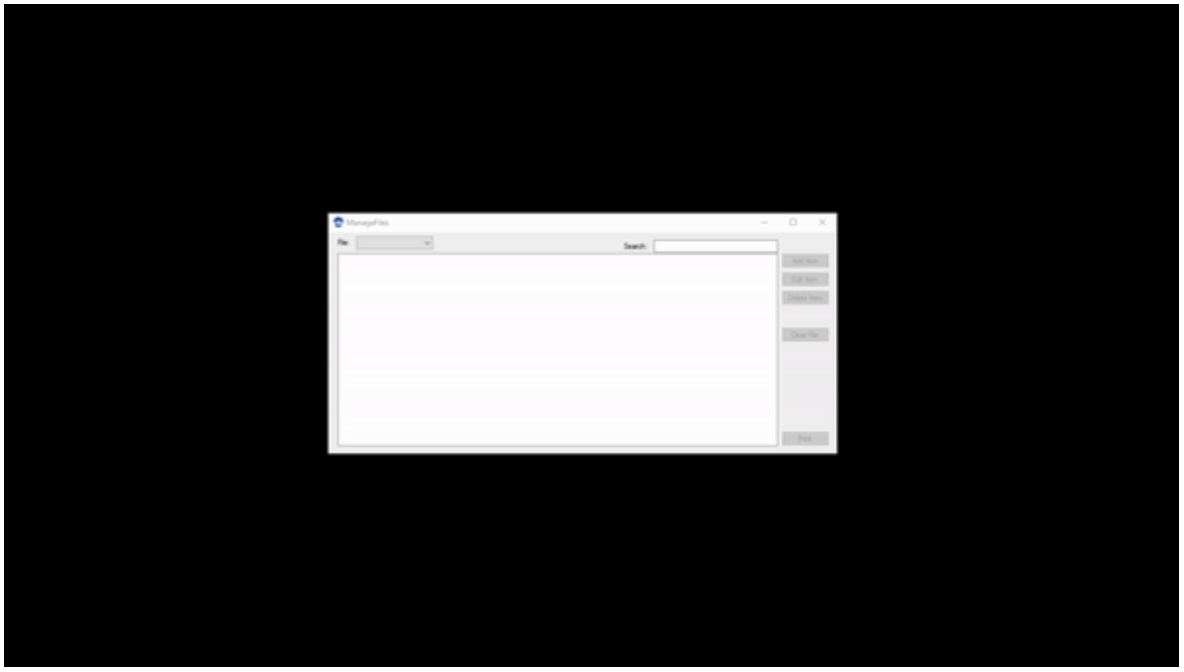




These screenshots show that the all items have been deleted from the customers.txt file, and are also no longer present in the **Manage Files** list either.

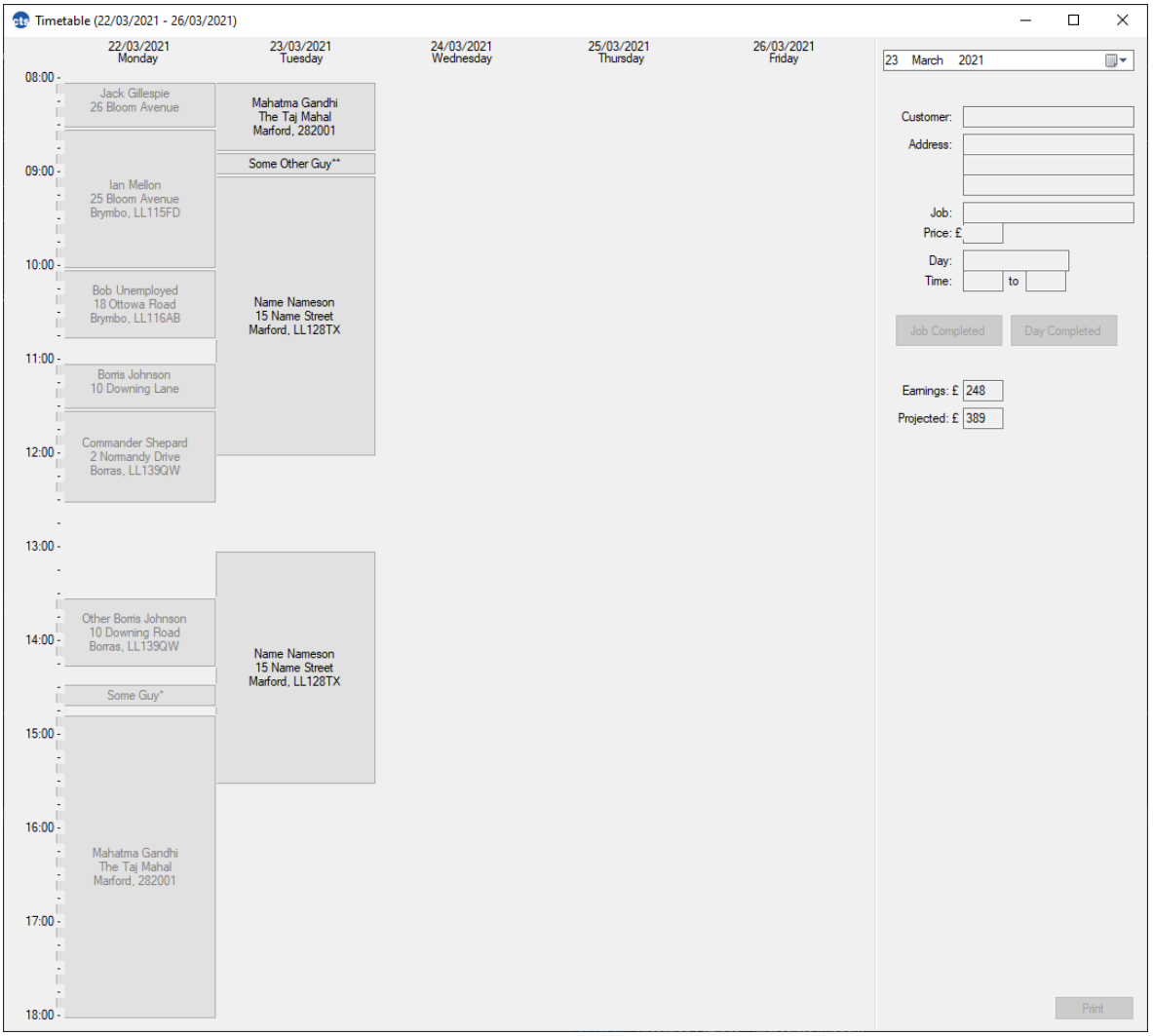
9	Resize form	Resize form	This will allow the user to make the form larger, allowing more items to be displayed at once. Whereas this option is disabled on the majority of the forms in this	Contents of form to change size along with the form.	**Contents of form changed size along with the form**

			system, it is not only allowed for thisform, but supported, as the contents of the form will scale with the size of the form.		

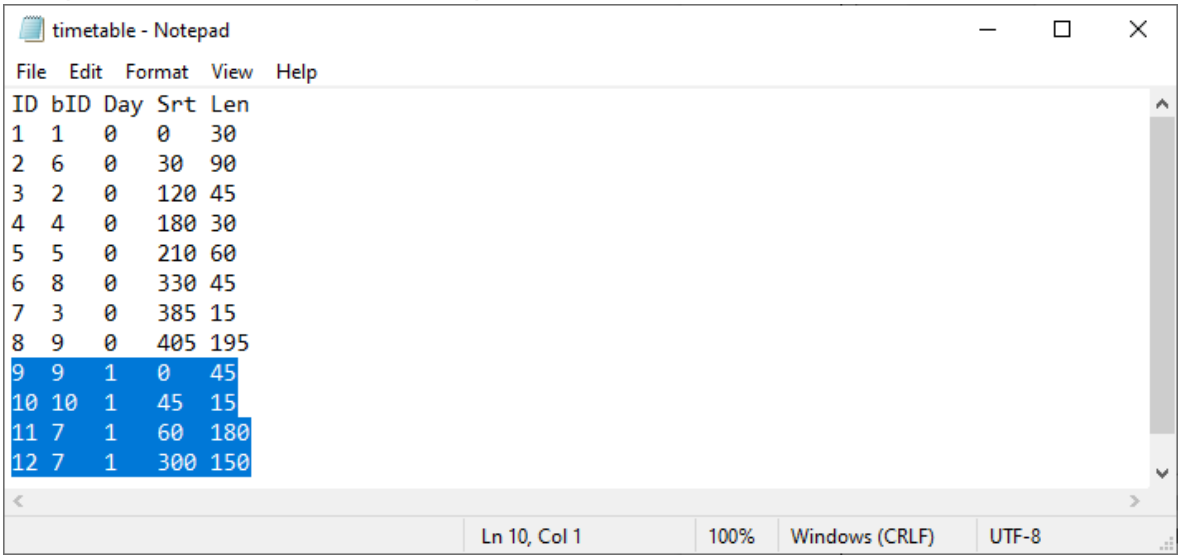


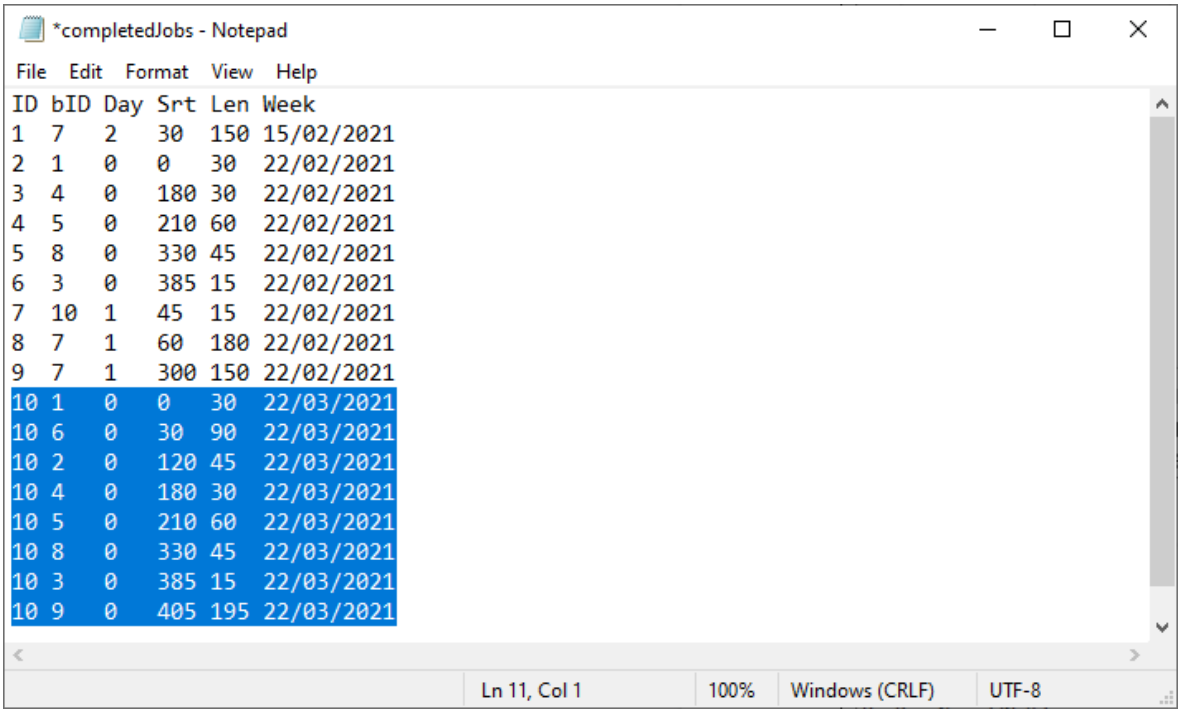
This screen capture shows the form changing size, and how the controls are properly anchored, and scale to fit the form.

Form: **View Timetable**					
Test No.	**Operation Tested**	**Event**	**Purpose**	**Expected Outcome**	**Observed Outcome**
1	Load timetable for current week	Open form	This will allow the user to view the timetable for the current week. The current week	Timetable successfully displayed.	**Timetable successfully displayed**



This screenshot was taken on a Tuesday. In black, the items of the current and future days are shown. In grey (in this case Monday's items) are the completed items, from the completedJobs.txt.

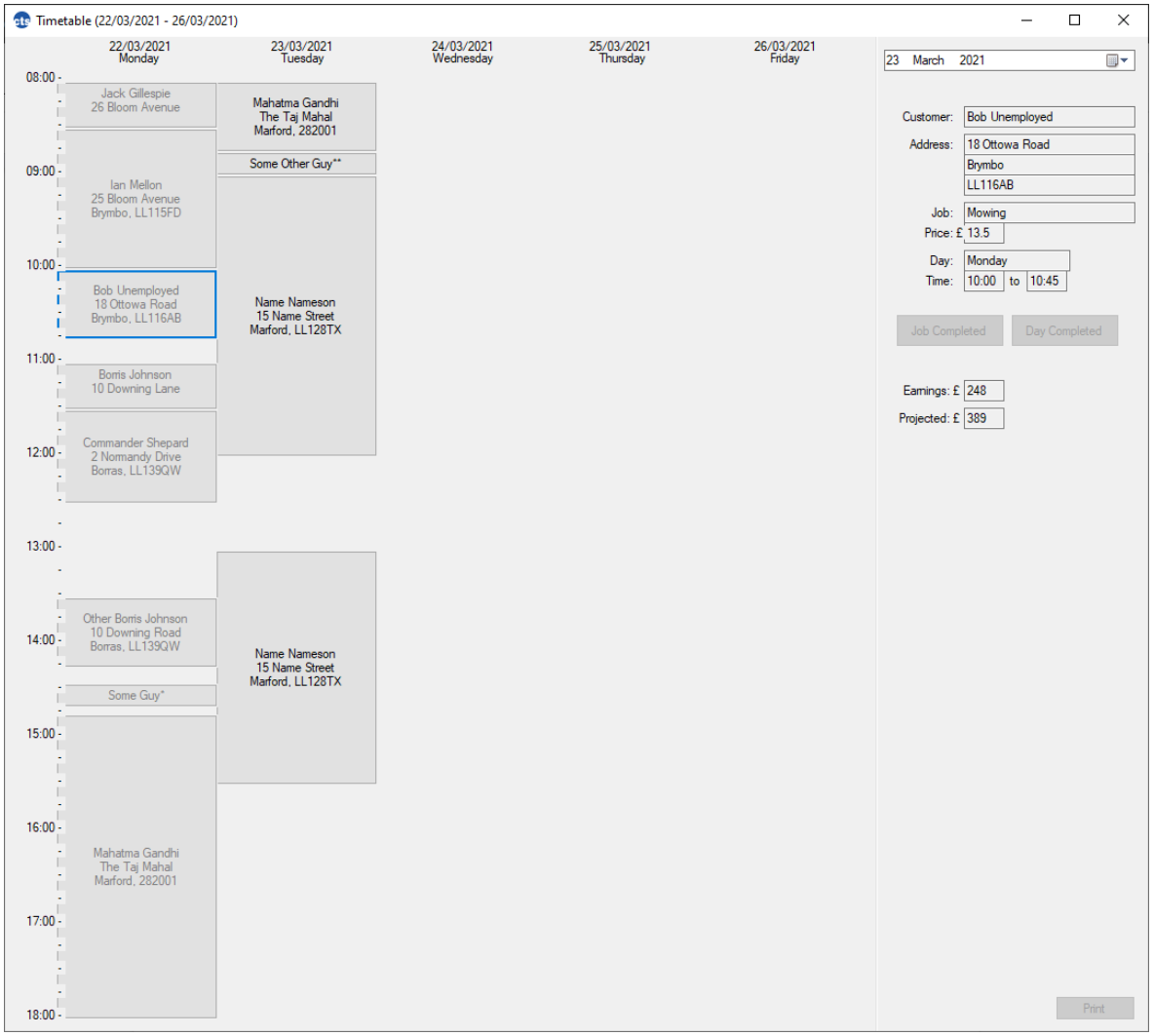




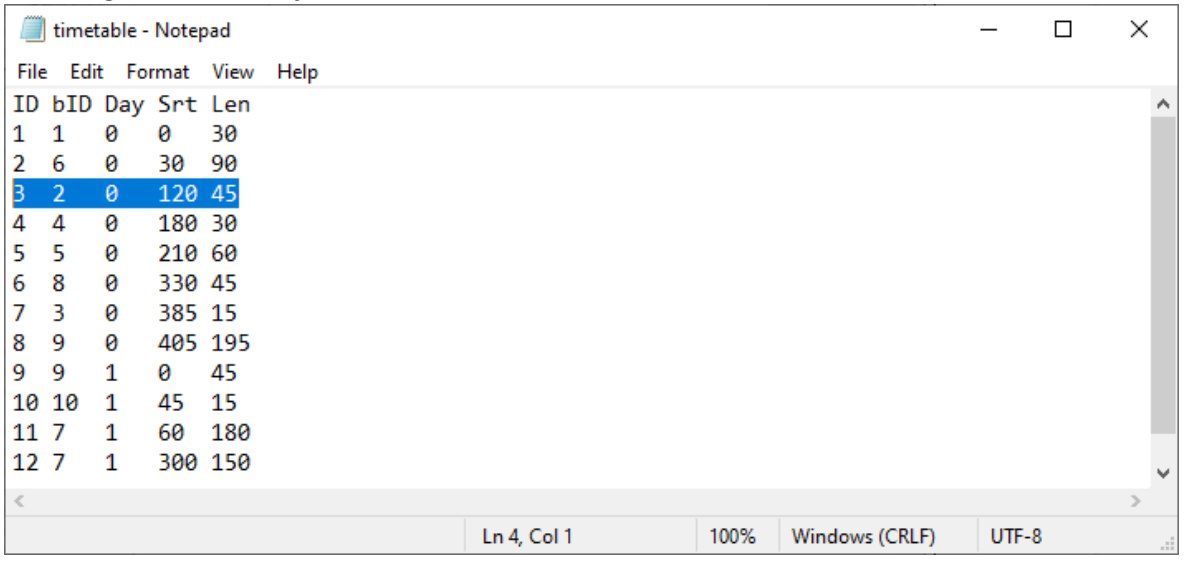
In the above screenshots, the data highlighted with blue is the data currently being displayed on the timetable.

This shows that the timetable does generate for the current week, using the timetable for future events, and only showing confirmed data from the past. The timetable boxes are all aligned with the days correctly, and do not overlap. The test below will use a specific example to ensure that the correct data is being displayed.

2	Load the correct data for the timetable for current week	Open form	Ensures the user is looking at the correct data, and not being misinformed.	Timetable uses correct data	**Timetable used correct data**

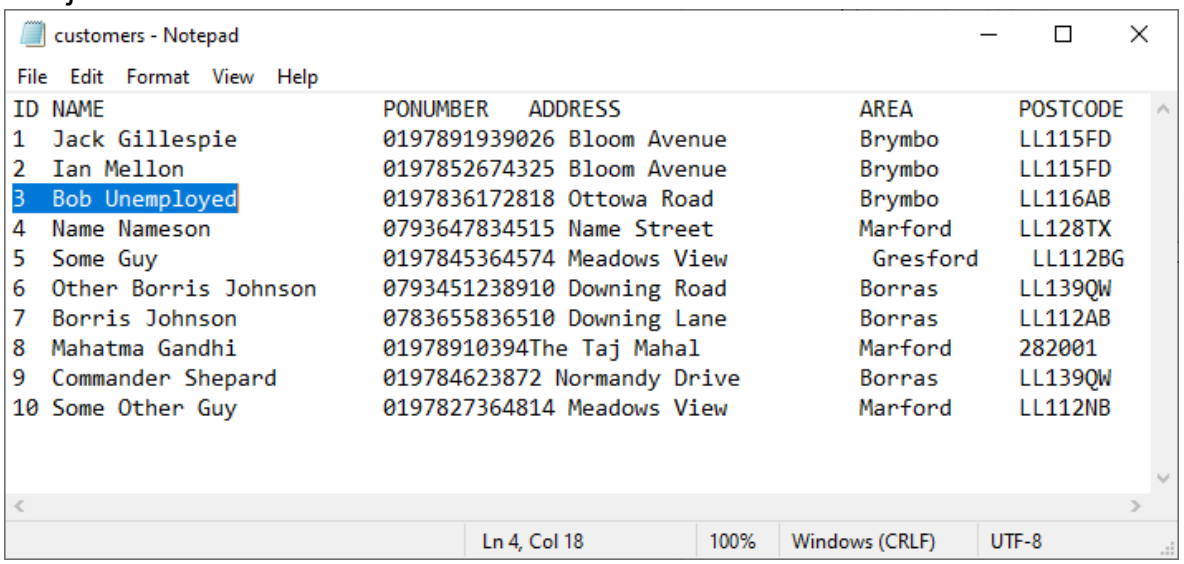


This screenshot shows that the selected item is for Bob Unemployed, Mowing, on Monday, at 10am to 10:45am.

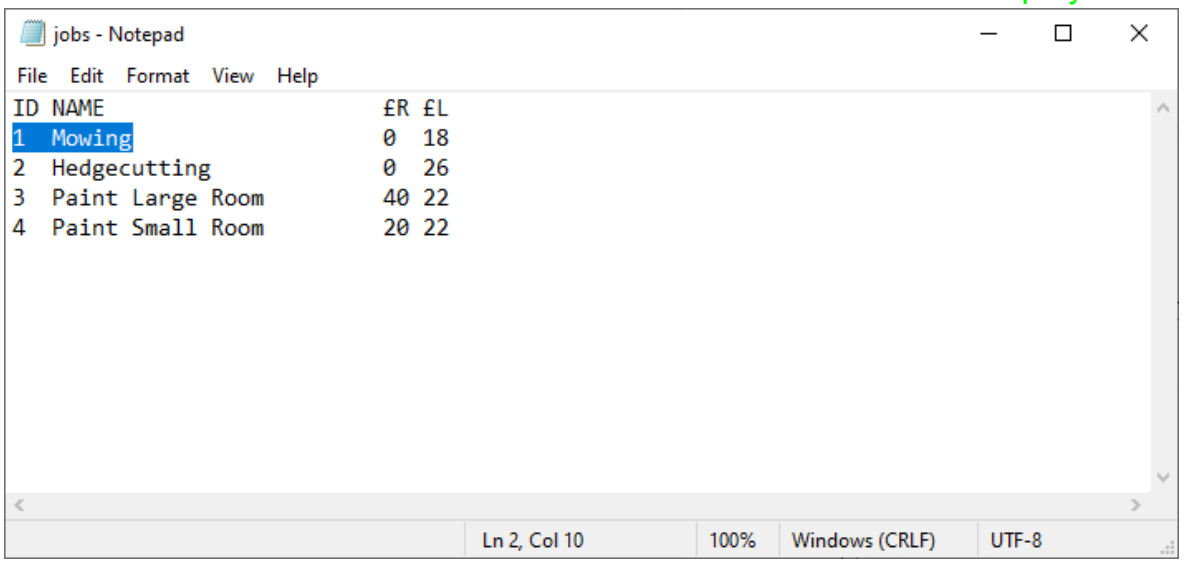


The above screenshot is the item selected. It has a start time of 8am + 120minutes = 10am, on day 0 (Monday) and length 45minutes until 10:45am.

The above screenshot shows the booking with ID 2, which has customer 3 and job 1.



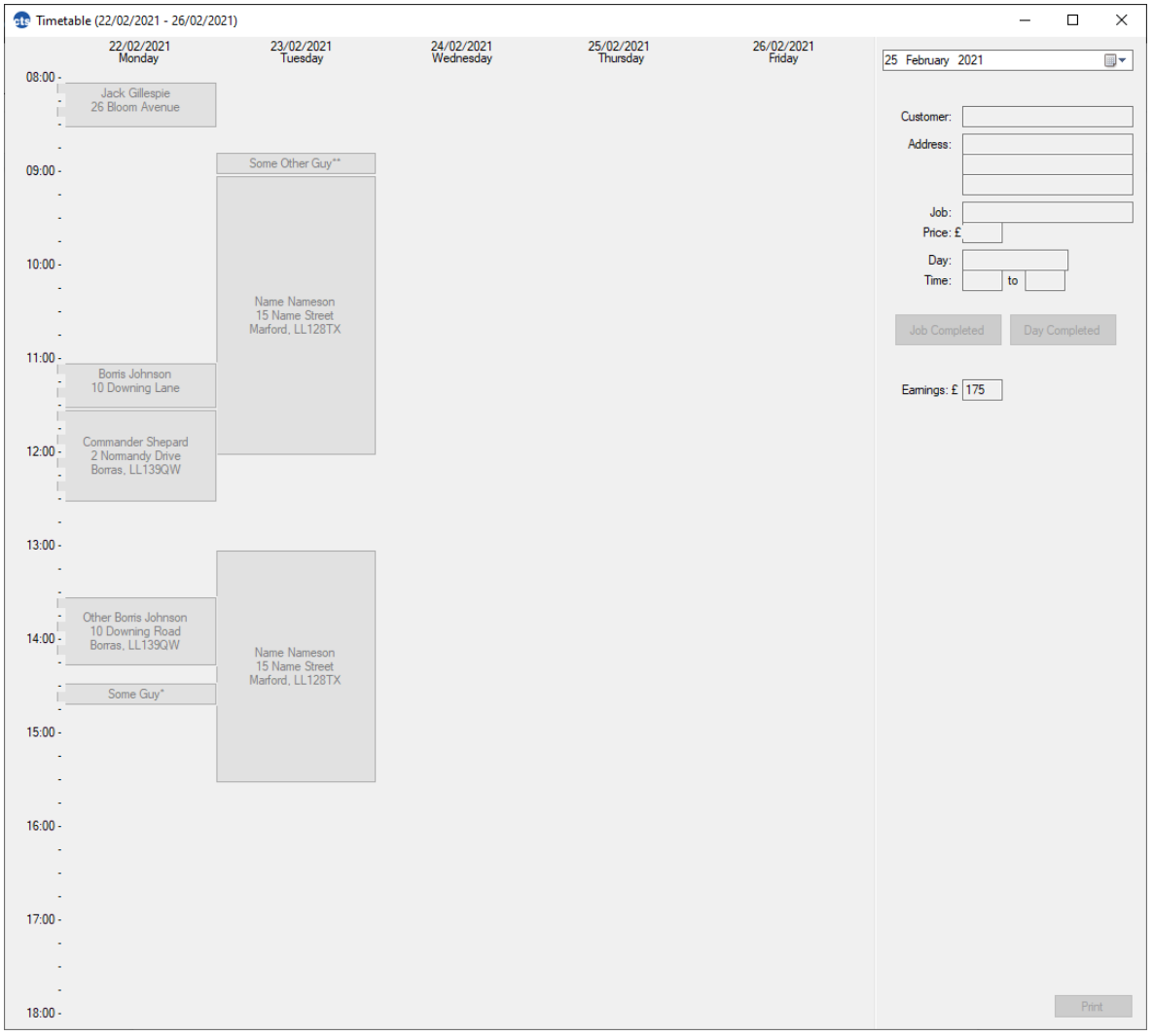
The above screenshot shows that customer 3 is indeed Bob Unemployed.

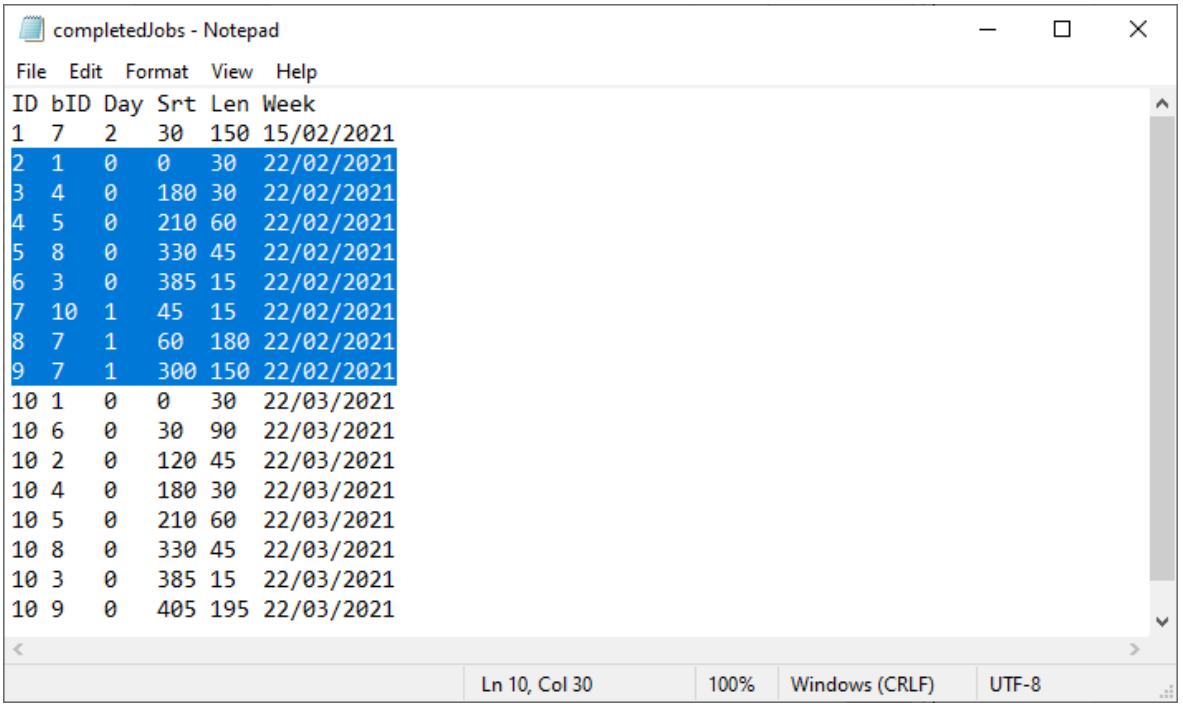


The above screenshot shows that job 1 is indeed Mowing.

This means that the timetable is displaying all of the correct information.The first screenshot displays this information in the top right of the page, as well as a more compressed form of it as the text of the button.

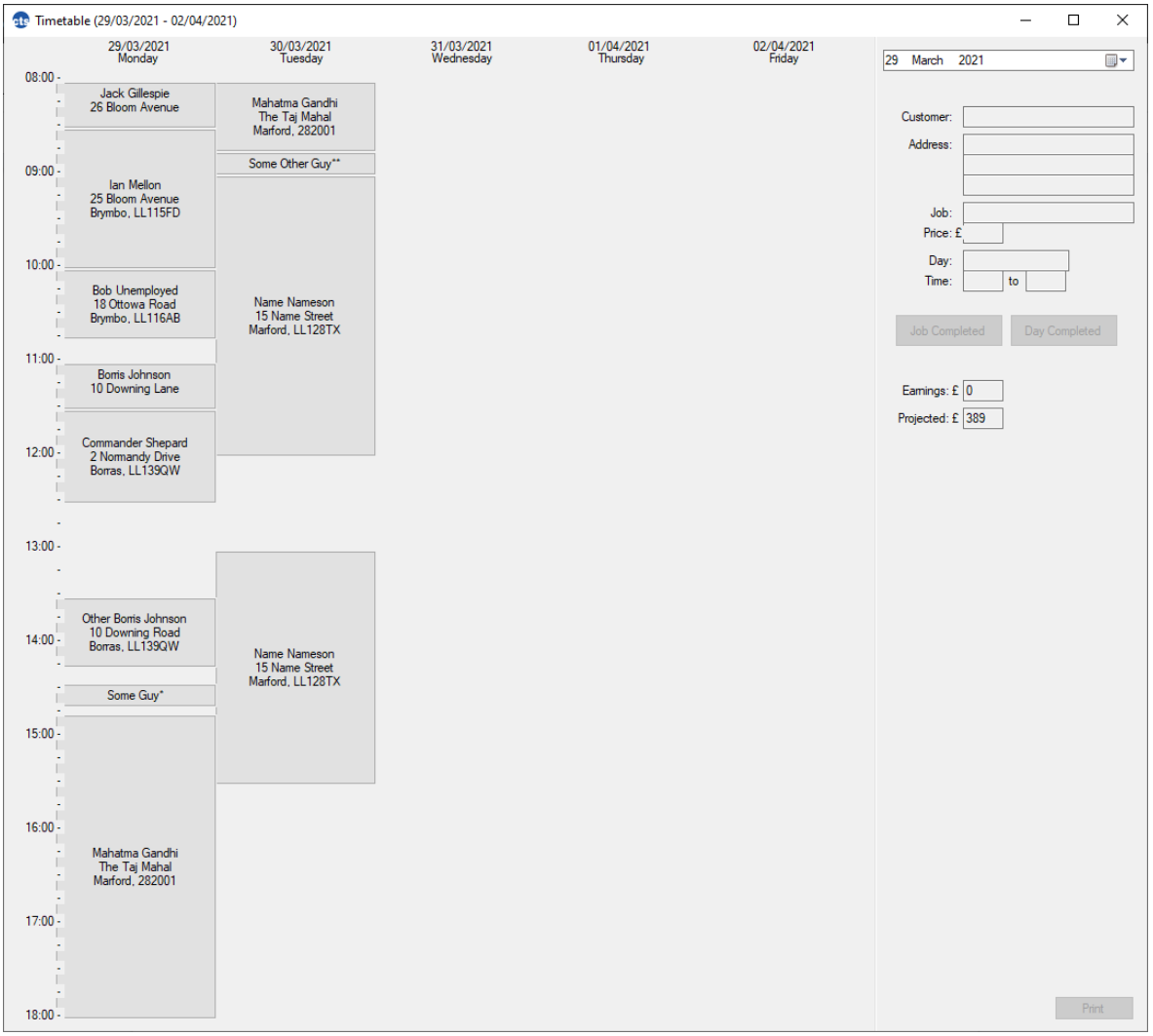
3	Load timetable for past week	Change date to a past week	This will allow the user to view the timetable for past weeks, which can be used to keep track of what jobs were done when, and for whom.	Timetable successfully displayed	**Timetable successfully displayed**

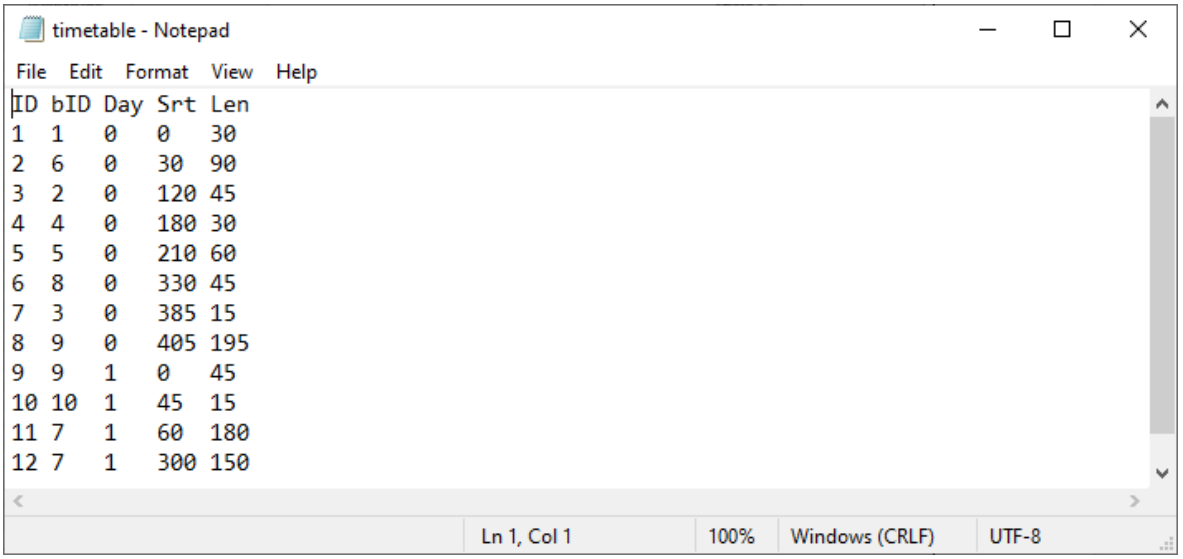




These screenshots show that the timetable can successfully display past items from the completedJobs.txt file, in this case loading all items from the week of 22/02/2021. As they are past items, the 'Job Completed' and 'Day Completed' buttons will be disabled, as these jobs are already completed and do not need completing again (the testing of these buttons is below).

4	Load timetable for future week	Change date to a future week	This will allow the user to view the timetable for future weeks. This timetable will be the same every week in the future, as it is a recurring weekly timetable.	Timetable successfully displayed	**Timetable successfully displayed**

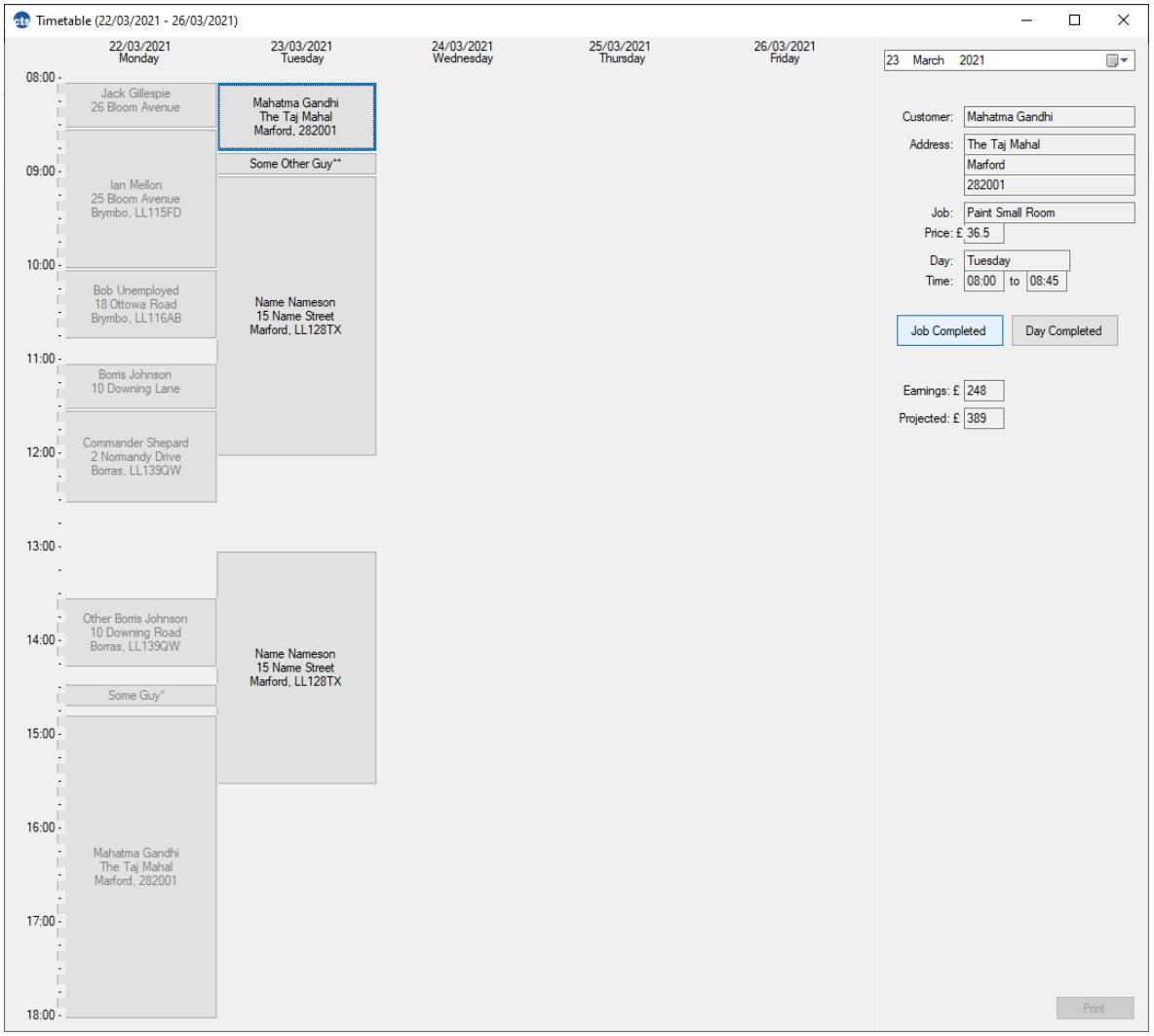


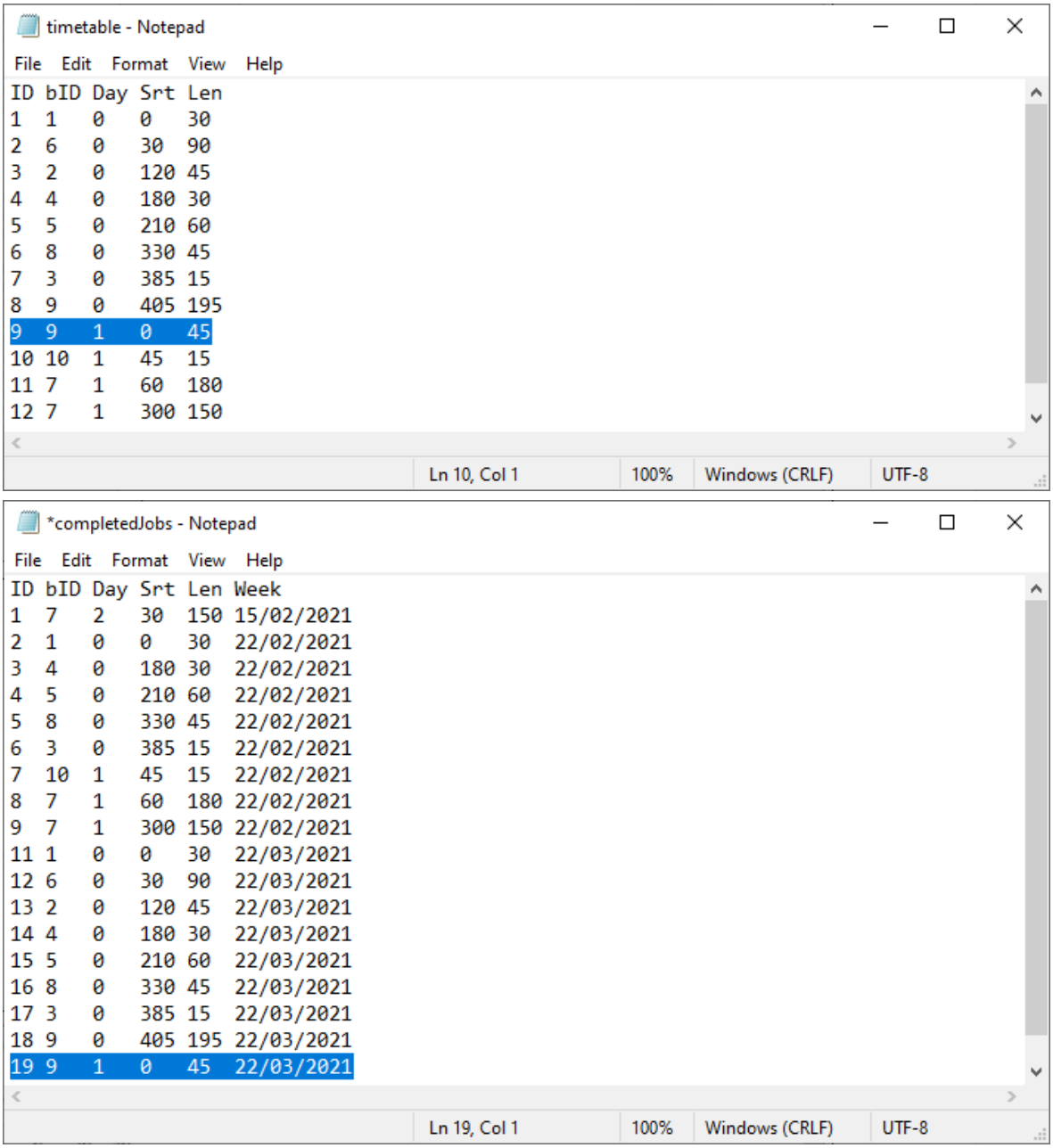


These screenshots show that the form can successfully display the timetable for future weeks using the timetable.txt file.

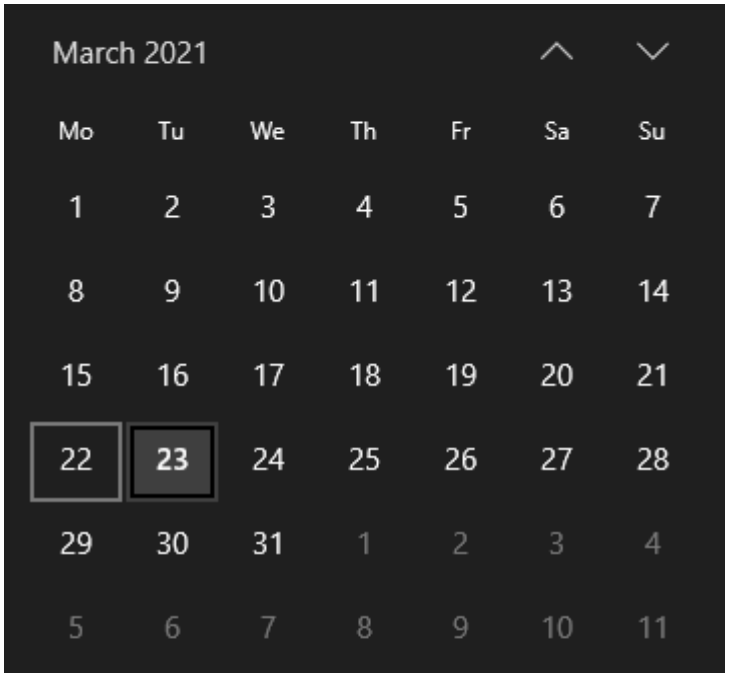
5	Mark job as completed	Click 'Job Complete	This will allow the user to mark an	Item successfu	**Item successf**

		d’	item as completed, storing it in the completedJobs.txt file	lly stored in the completed Jobs.txtfile	**ully stored in the completedJobs.txt file**

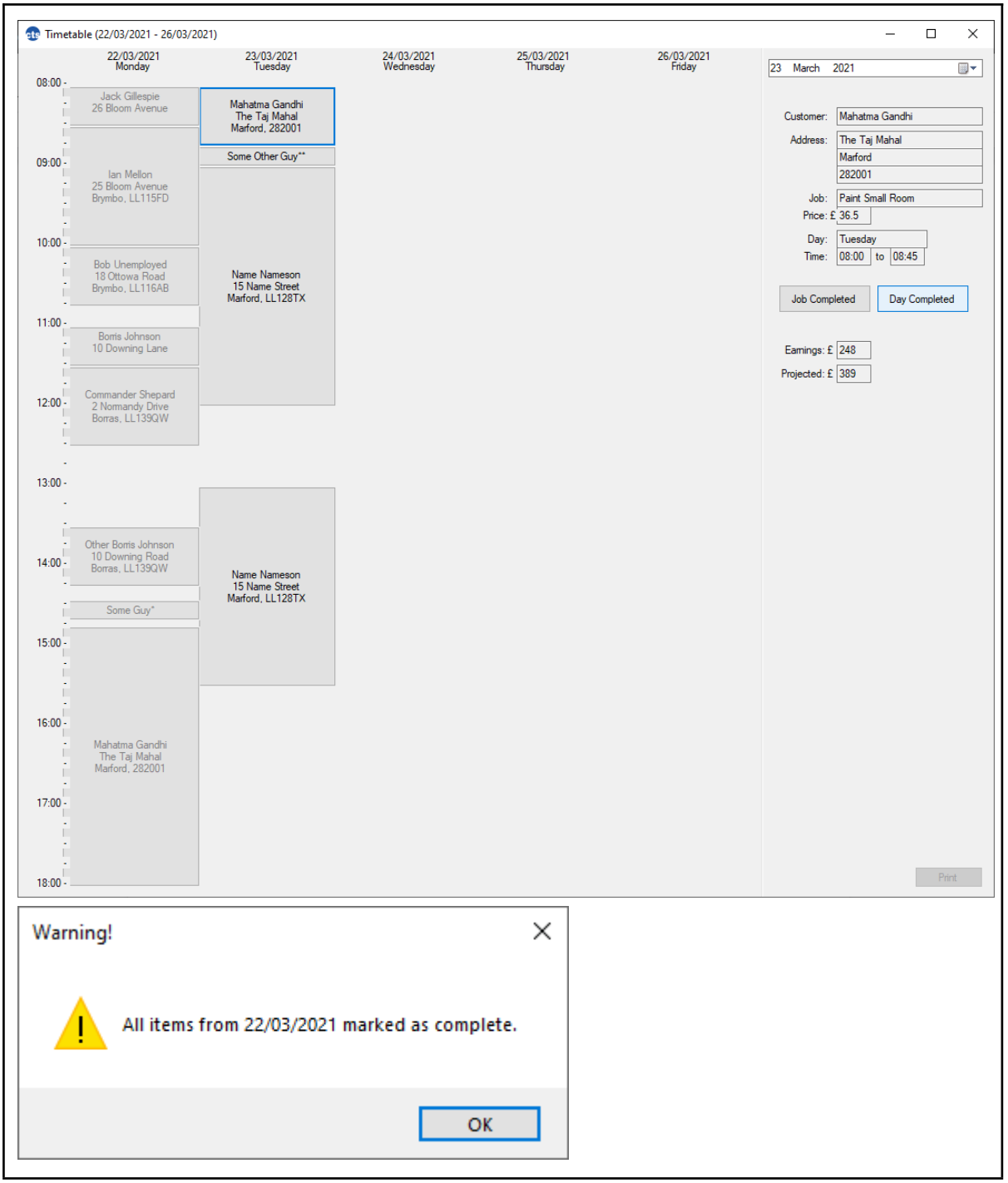


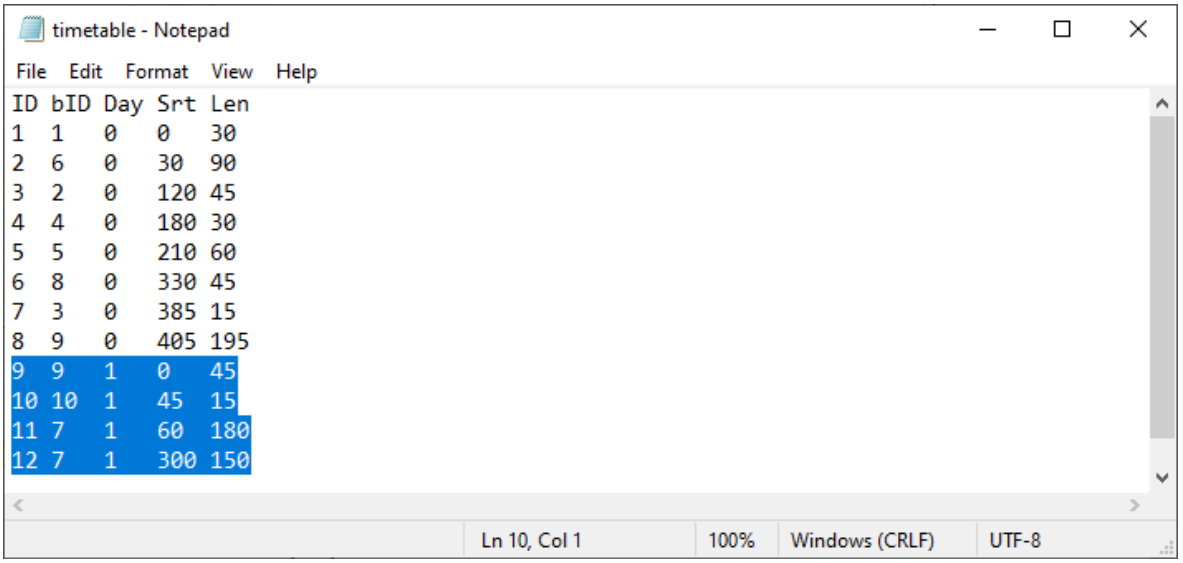


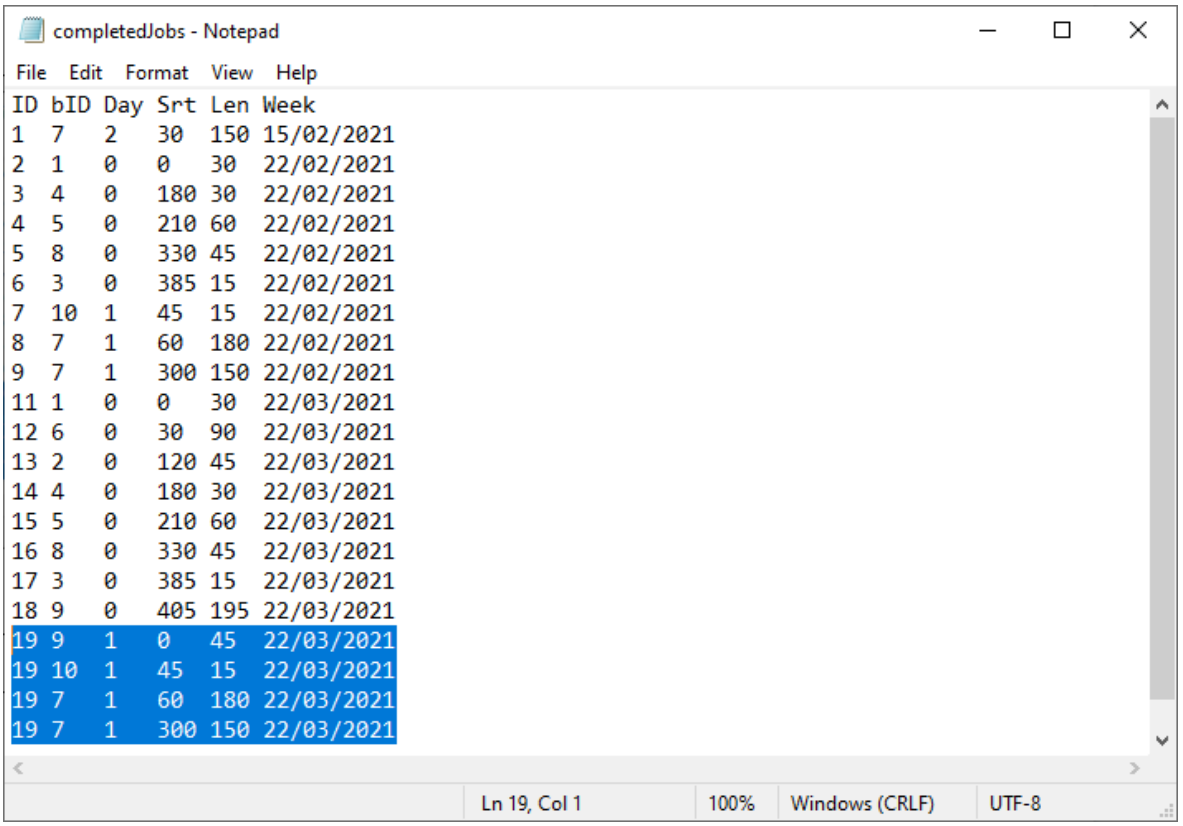
These screenshots show that item 9 from timetable.txt has been added to completeddJobs.txt (with a new ID of 19). The bID, Day, StartTime, Length are all the same, and the week is that of the current week.



6	Mark all jobs in same day as completed	Click ‘Day Completed’	This will allow the user to mark multiple items as completed, storing it in the completedJobs.txt file	Items successfully stored in the completedJobs.txtfile	**Items successfully stored in the completedJobs.txt file.** **IDs not unique.**


Timetable (22/03/2021 - 26/03/2021)
22/03/2021 Monday
23/03/2021 Tuesday
24/03/2021 Wednesday
25/03/2021 Thursday
26/03/2021 Friday
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
Jack Gillespie
26 Bloom Avenue
Ian Mellon
25 Bloom Avenue
Brymbo, LL115FD
Bob Unemployed
18 Ottowa Road
Brymbo, LL116AB
Borris Johnson
10 Downing Lane
Commander Shepard
2 Normandy Drive
Borras, LL139QW
Other Borris Johnson
10 Downing Road
Borras, LL139QW
Some Guy*
Mahatma Gandhi
The Taj Mahal
Marford, 282001
Mahatma Gandhi
The Taj Mahal
Marford, 282001
Some Other Guy**
Name Nameson
15 Name Street
Marford, LL128TX
Name Nameson
15 Name Street
Marford, LL128TX
23 March 2021
Customer: Mahatma Gandhi
Address: The Taj Mahal
Marford
282001
Job: Paint Small Room
Price: £ 36.5
Day: Tuesday
Time: 08:00 to 08:45
Job Completed
Day Completed
Earnings: £ 248
Projected: £ 389
Print
Warning!
All items from 22/03/2021 marked as complete.
OK





This process is highly similar to the job completed process, however it saves multiple items at once. These screenshots show that items 9, 10, 11 and 12 from timetable.txt have been added to completeddJobs.txt. **However**, all of these items have the same ID of 19. This means the primary key of this file is not unique, which can result in errors as the program will not be able to uniquely differentiate the items. This is caused by an error in the code that generates the ID.

6.1	same as above	same as above	same as above	Unique IDs	**Unique IDs**

```
39      Function GenerateID()
40
41          fileContents = File.ReadAllLines(Dir$(fileName))
42
43          'GENERATE ID
44          Dim highestValue As Integer, currentValue As Integer
45
46          For x = 1 To fileContents.Count - 1
47
48              currentValue = Trim(Mid(fileContents(x), 1, 3))
49
50              If currentValue > highestValue Then
51                  highestValue = currentValue
52              End If
53
54          Next
55
56          ID = highestValue + 1
57
58          Return ID
59
60      End Function
```

Added line 41, to ensure the file contents variable are updated as the file is updated, and so the IDs are unique.

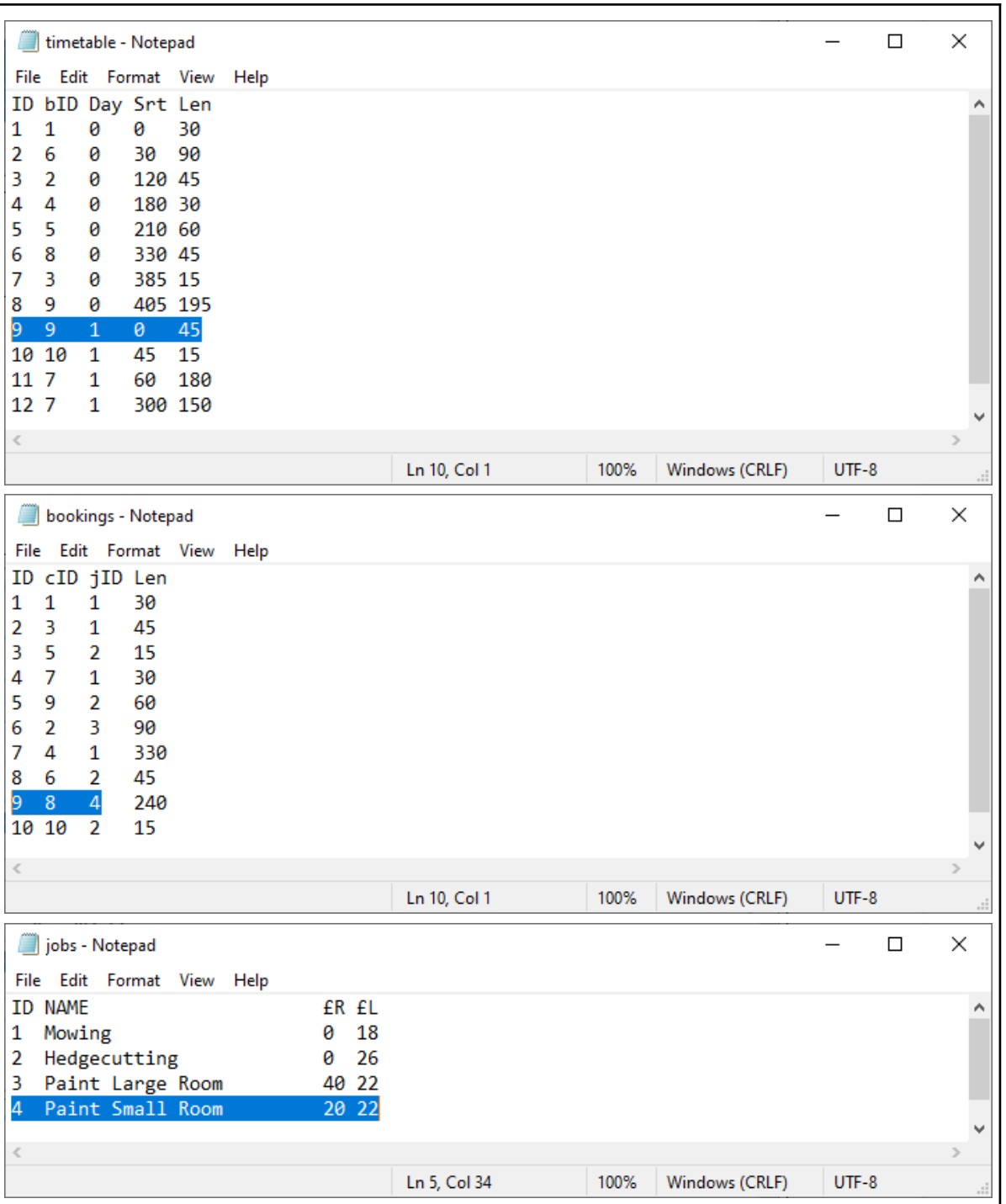
*completedJobs - Notepad

File Edit Format View Help

ID	bID	Day	Srt	Len	Week
1	7	2	30	150	15/02/2021
2	1	0	0	30	22/02/2021
3	4	0	180	30	22/02/2021
4	5	0	210	60	22/02/2021
5	8	0	330	45	22/02/2021
6	3	0	385	15	22/02/2021
7	10	1	45	15	22/02/2021
8	7	1	60	180	22/02/2021
9	7	1	300	150	22/02/2021
11	1	0	0	30	22/03/2021
12	6	0	30	90	22/03/2021
13	2	0	120	45	22/03/2021
14	4	0	180	30	22/03/2021
15	5	0	210	60	22/03/2021
16	8	0	330	45	22/03/2021
17	3	0	385	15	22/03/2021
18	9	0	405	195	22/03/2021
19	9	1	0	45	22/03/2021
20	10	1	45	15	22/03/2021
21	7	1	60	180	22/03/2021
22	7	1	300	150	22/03/2021

Ln 19, Col 1 100% Windows (CRLF) UTF-8

7	Calculate price of job	Select item	The system will calculate the price of the job.	36.5 see below	**36.5**



timetable - Notepad

File Edit Format View Help

ID	bID	Day	Srt	Len
1	1	0	0	30
2	6	0	30	90
3	2	0	120	45
4	4	0	180	30
5	5	0	210	60
6	8	0	330	45
7	3	0	385	15
8	9	0	405	195
9	9	1	0	45
10	10	1	45	15
11	7	1	60	180
12	7	1	300	150

Ln 10, Col 1 | 100% | Windows (CRLF) | UTF-8

bookings - Notepad

File Edit Format View Help

ID	cID	jID	Len
1	1	1	30
2	3	1	45
3	5	2	15
4	7	1	30
5	9	2	60
6	2	3	90
7	4	1	330
8	6	2	45
9	8	4	240
10	10	2	15

Ln 10, Col 1 | 100% | Windows (CRLF) | UTF-8

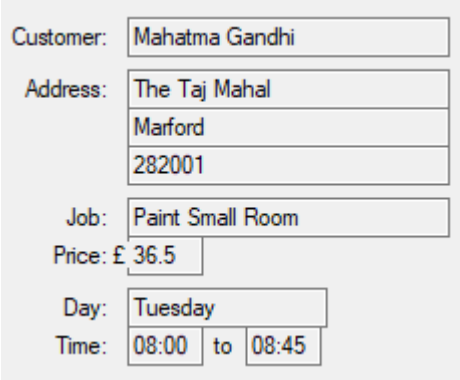
jobs - Notepad

File Edit Format View Help

ID	NAME	£R	£L
1	Mowing	0	18
2	Hedgecutting	0	26
3	Paint Large Room	40	22
4	Paint Small Room	20	22

Ln 5, Col 34 | 100% | Windows (CRLF) | UTF-8

price = resource price + (hourly rate * hours)
= 20 + 22(0.75) = 36.5



This screenshot shows that the calculated price of this job is £36.50, which shows it is correctly generating the prices of the individual jobs.

8	Calculate price of week	Load form	The system will calculate the sum price of the jobs in any given week. It will have a projected sum (amount that will be earnt if all tasks are completed) and the actual earnings (money from completed jobs).	248 389 see below	**248** **389**

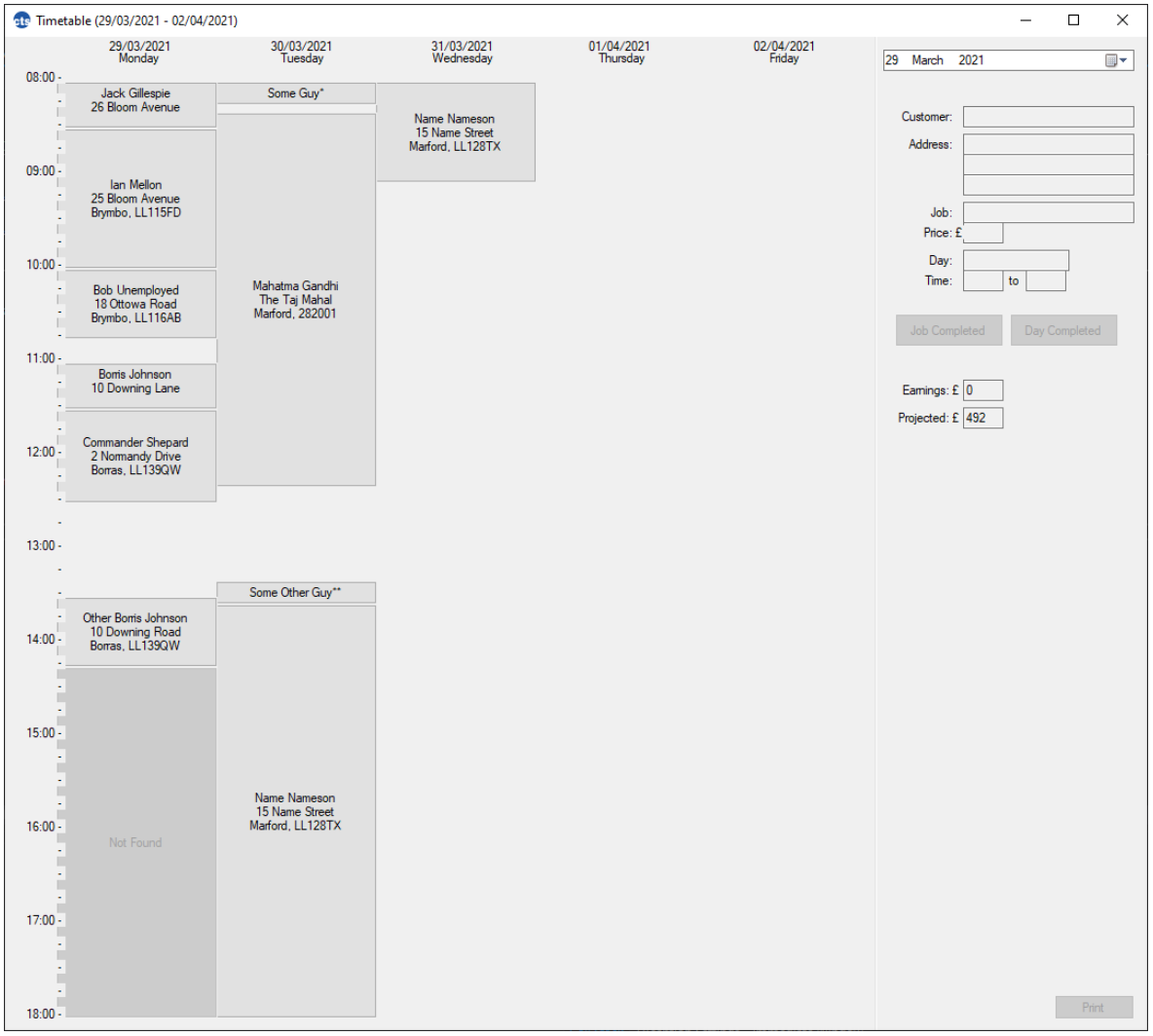
price = SUM(resource price + (hourly rate * hours))
= 9 + 73 + 13.5 + 9 + 26 + 19.5 + 6.5 + 91.5
= 248

price = SUM(resource price + (hourly rate * hours))
= 9 + 73 + 13.5 + 9 + 26 + 19.5 + 6.5 + 91.5 + 36.5 + 6.5 + 54 + 44
= 389

This shows that the timetable successfully calculates both the projected and real earnings of any given week, and that it's result is accurate.

9	Display timetable after deleting vital	Delete item, load form	It is important that the system will not break if data is lost, as the	"Not Found"	**"Not Found"**

	information		user may accidentally delete important information, or, the file may corrupt.		

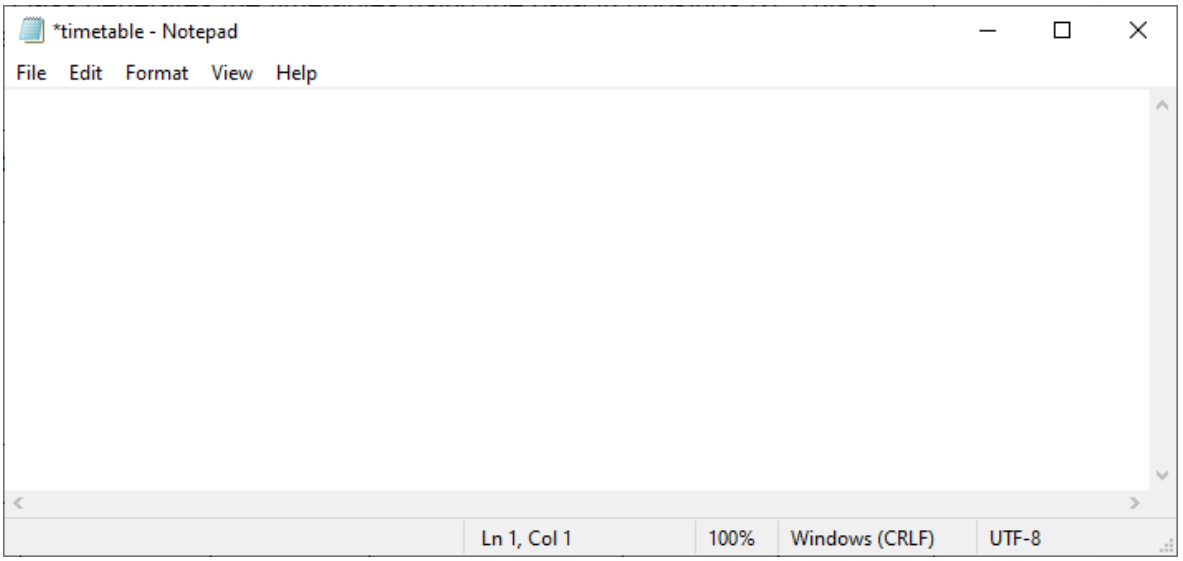


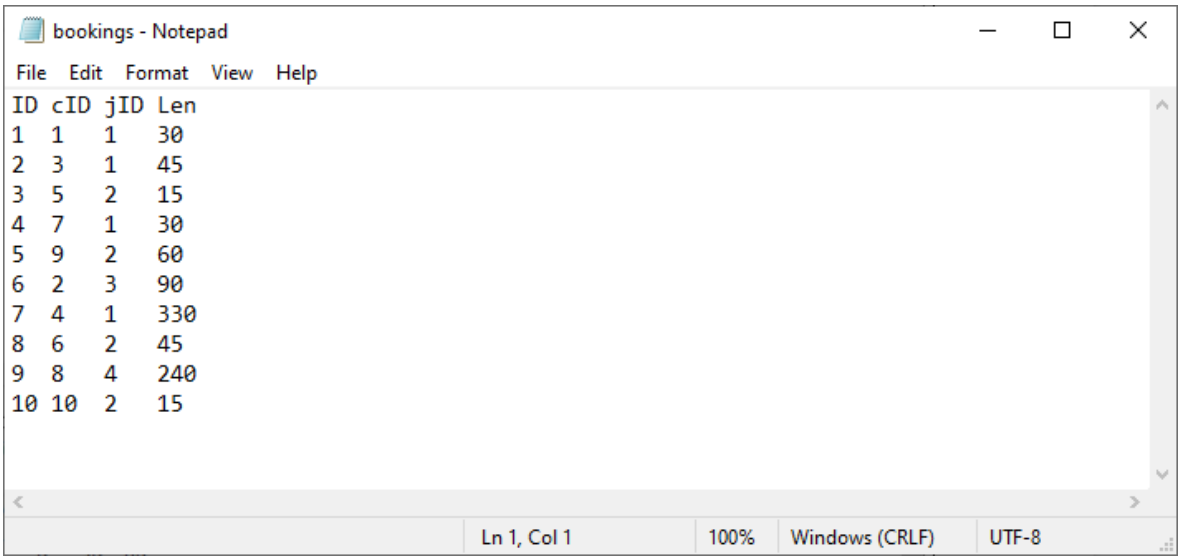
This screenshot shows that the timetable item (last item on Monday) where the booking has since been deleted is now replaced by an error message. This will only ever occur for past weeks, as when future bookings are deleted, the timetable is regenerated to compensate for this.

Timetable Calculations Class

This class generates the timetables using the data in bookings.txt. This is done through a series of sorts, and then splitting the items over the course of 5 days. It is called by **Add Booking** and **Manage Files** so that a new

timetable is generated whenever bookings are added, edited, or deleted.					
Test No.	**Operation Tested**	**Event**	**Purpose**	**Expected Outcome**	**Observed Outcome**
1	Generate timetable	Triggered by from	This will generate a table when the contents of the bookings file is altered.	Timetable successfully generated	**Timetable successfully generated**





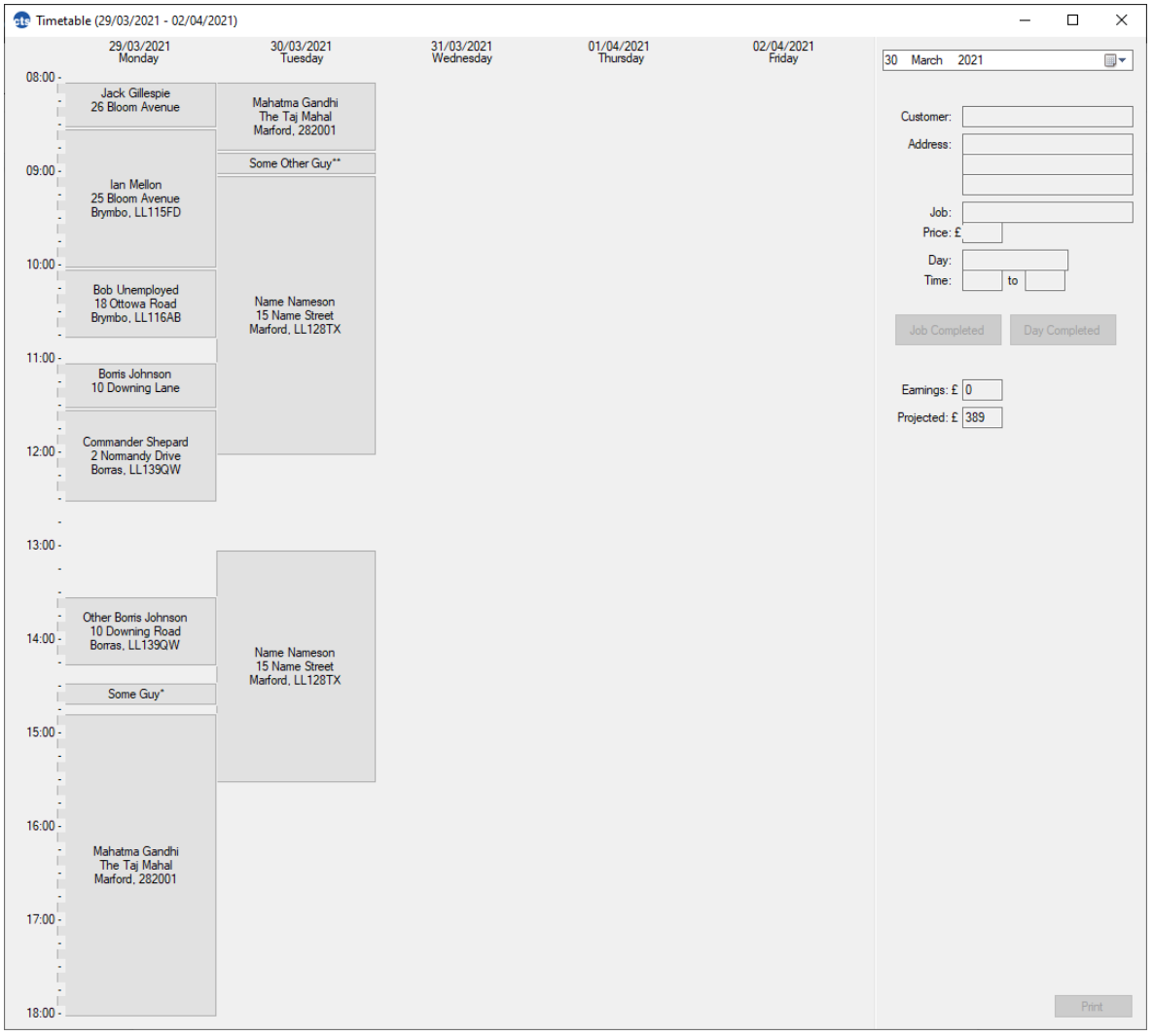
timetable - Notepad

File Edit Format View Help

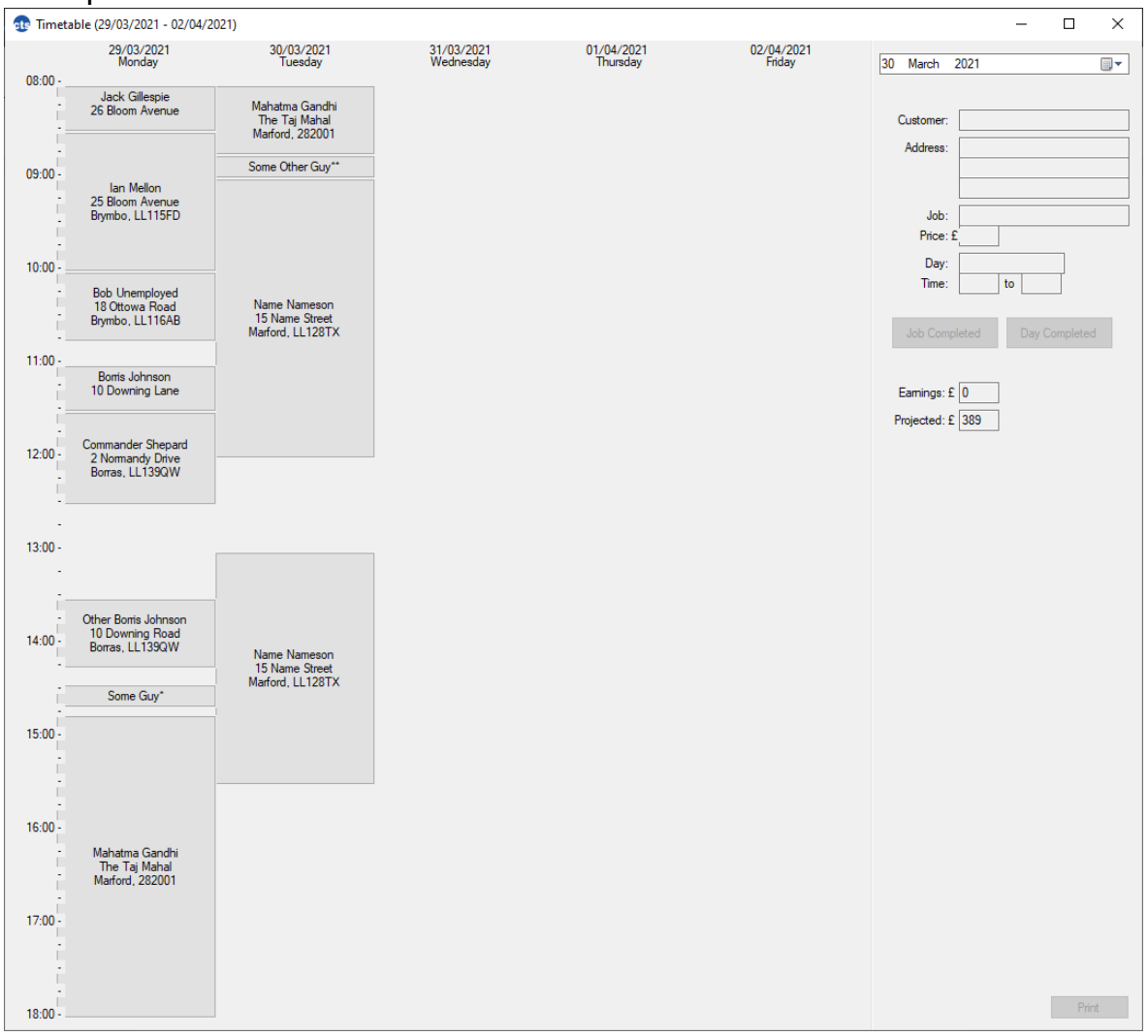
```
ID bID Day Srt Len
1  1   0   0   30
2  6   0   30  90
3  2   0   120 45
4  4   0   180 30
5  5   0   210 60
6  8   0   330 45
7  3   0   385 15
8  9   0   405 195
9  9   1   0   45
10 10  1   45  15
11 7   1   60  180
12 7   1   300 150
```

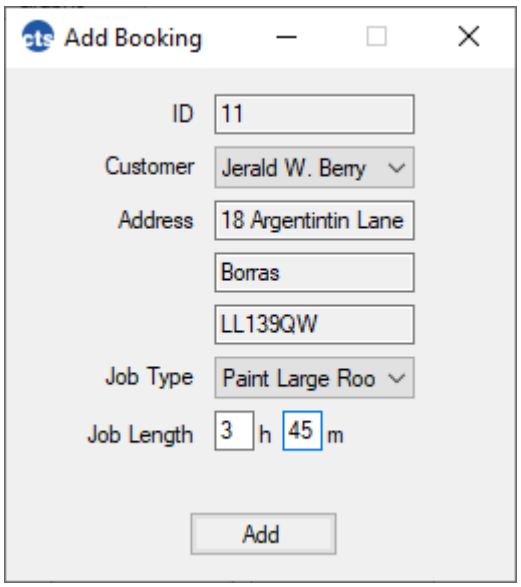
Ln 1, Col 1 100% Windows (CRLF) UTF-8

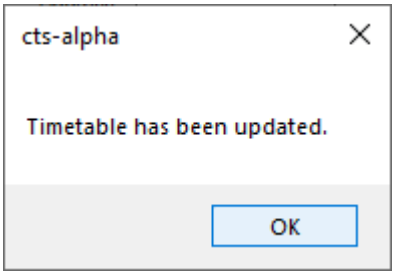
This shows the timetable that is generated from this example of booking data. A visualisation of the timetable can be seen in the screenshot below. An overview of the timetable generation process can be seen in the design document.

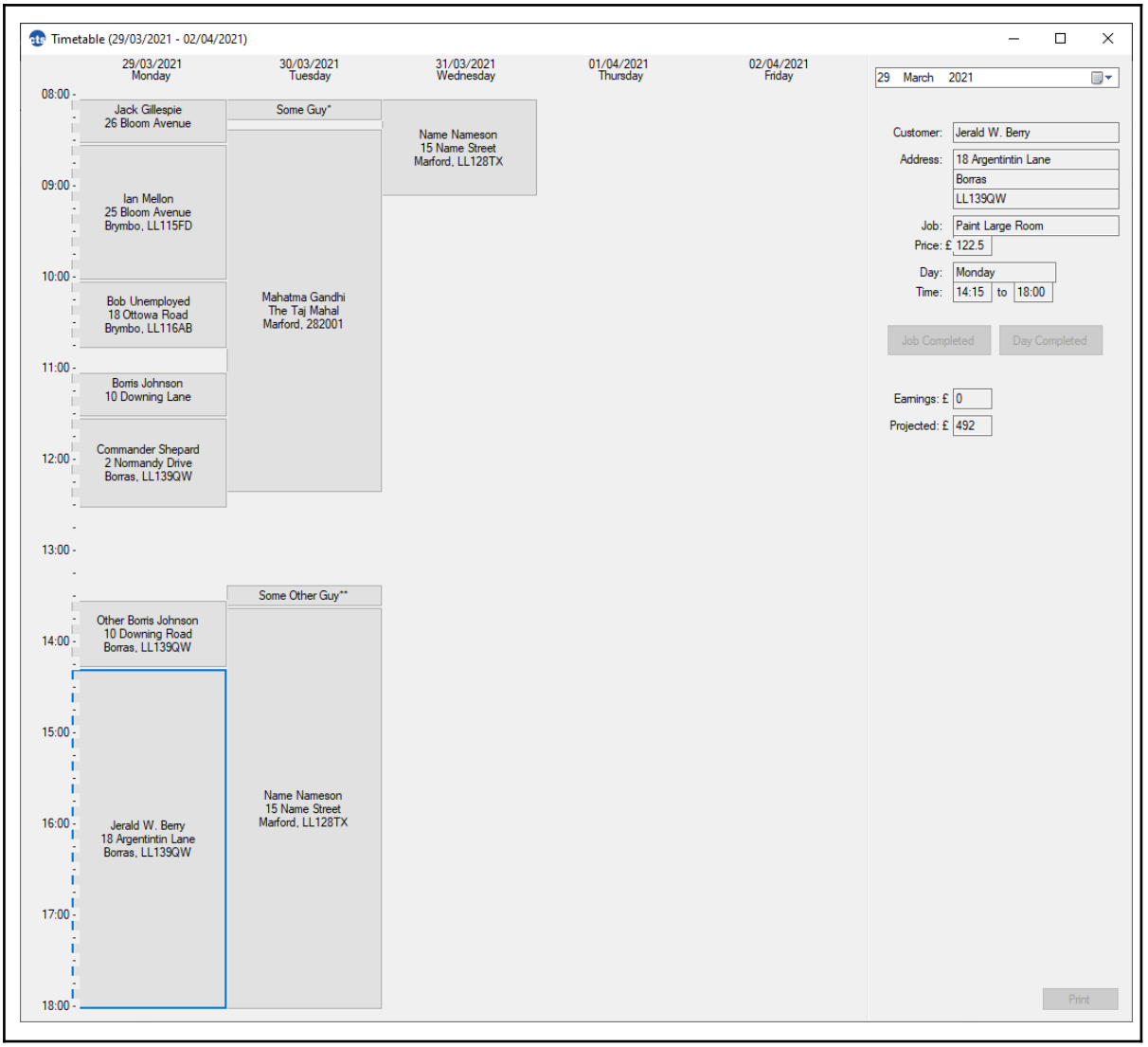


Example 2:

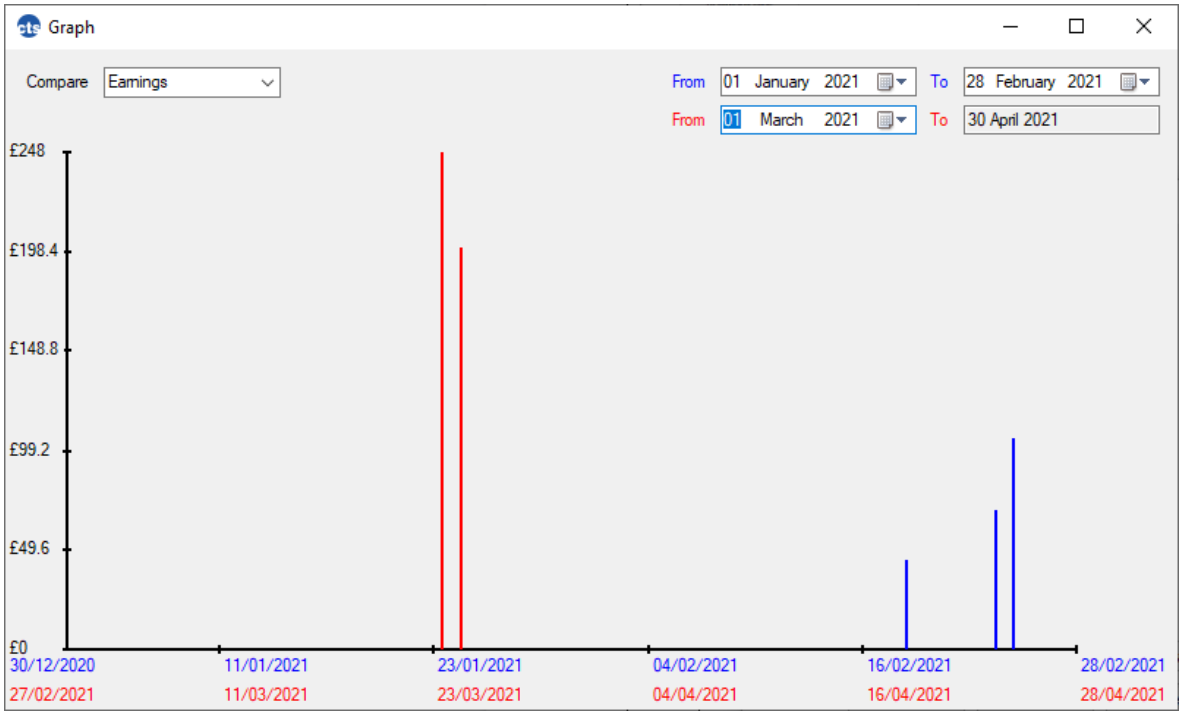

Timetable (29/03/2021 - 02/04/2021)
29/03/2021 Monday
30/03/2021 Tuesday
31/03/2021 Wednesday
01/04/2021 Thursday
02/04/2021 Friday
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
Jack Gillespie
26 Bloom Avenue
Ian Mellon
25 Bloom Avenue
Brymbo, LL115FD
Bob Unemployed
18 Ottowa Road
Brymbo, LL116AB
Borris Johnson
10 Downing Lane
Commander Shepard
2 Normandy Drive
Borras, LL139QW
Other Borris Johnson
10 Downing Road
Borras, LL139QW
Some Guy*
Mahatma Gandhi
The Taj Mahal
Marford, 282001
Some Other Guy**
Name Nameson
15 Name Street
Marford, LL128TX
Name Nameson
15 Name Street
Marford, LL128TX
Mahatma Gandhi
The Taj Mahal
Marford, 282001
30 March 2021
Customer:
Address:
Job:
Price: £
Day:
Time:
to
Job Completed
Day Completed
Earnings: £ 0
Projected: £ 389
Print


Add Booking
ID 11
Customer Jerald W. Berry
Address 18 Argentintin Lane
Borras
LL139QW
Job Type Paint Large Roo
Job Length 3 h 45 m
Add


cts-alpha
Timetable has been updated.
OK

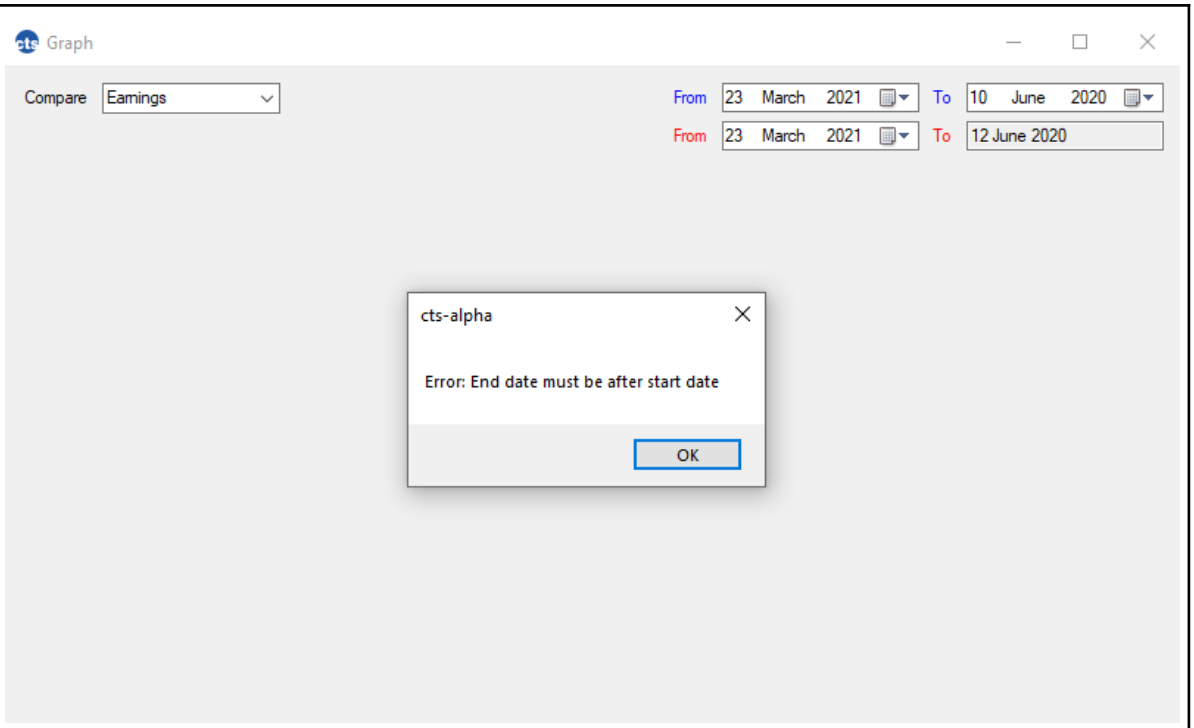


Form: **Graph**					
Test No.	**Operation Tested**	**Event**	**Purpose**	**Expected Outcome**	**Observed Outcome**
1	Generate graph comparing earnings from Jan1st - Feb28th and Mar1st - April30th	Select the aforementioned dates	This will allow the user to see a visualisation of their earnings over a period of time, and compare two time periods.	Graph successfully generated	**Graph successfully generated**



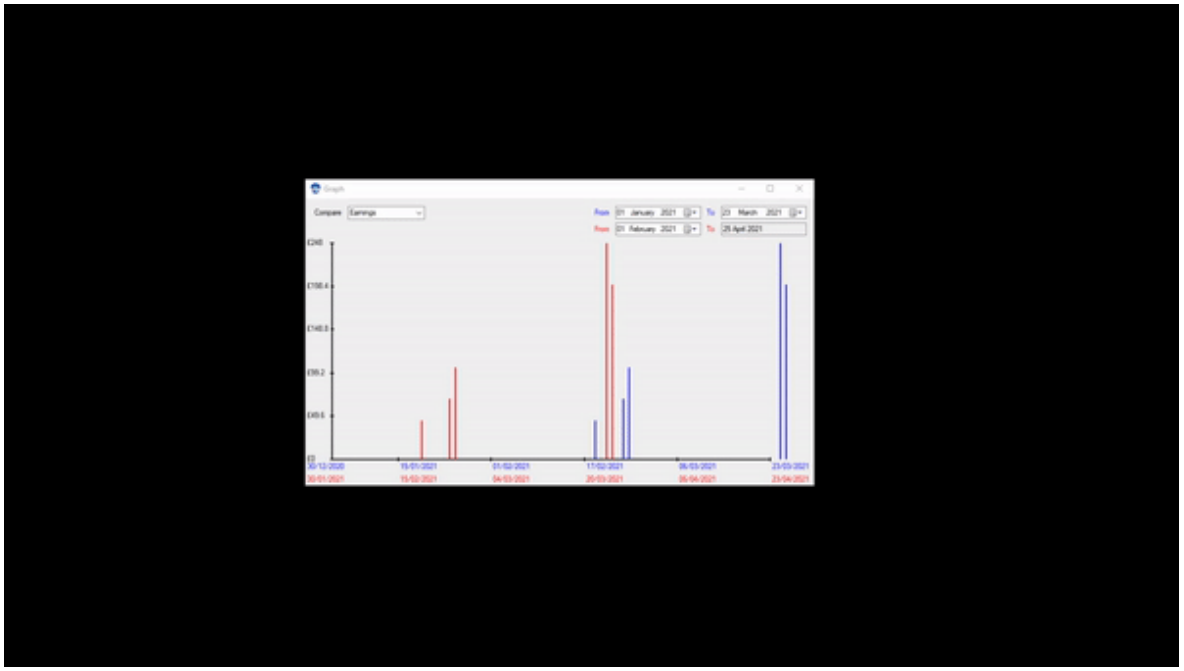
This screenshot shows a graph of earnings between Jan1st - Feb28th and Mar1st - April30th. The earnings for these graphs (y axis) is calculated using the same function as the **View Timetable** function, and so does not require testing as it is known to be correct.

2	Generate graph with the end date being before the start date	Select an end date that is before the start date.	This could be an easily done error by the user, as they may accidentally select a date from the past for the end date that is before the start date. This could potentially break calculations, and so needs a preventative feature.	Error message	**“Error: End date must be after start date”**



This screenshot shows that the preventative feature works, and that end dates cannot be from before the start dates, meaning there shouldn't be any negative values for the graph to process, which would break it.

3	Resize form	Resize form	This will allow the user to make the form larger, allowing more items to be displayed at once. Whereas this option is disabled on the majority of the forms in this system, it is not only allowed for thisform, but supported, as the contents of the form will scale with the size of the form.	Contents of form to change size along with the form.	**Contents of form changed size along with the form**



This screen capture shows the form changing size, and how the controls are properly anchored, and scale to fit the form.

Developmental Testing

During development, testing was an on-going process as features were added and altered. Below is a list of some of the errors encountered during this process, and the solutions to them.

16/12/2020	Route finding algorithm sorting incorrectly

input - Notepad

File Edit Format View Help

ID	AREA	POSTCODE
1	Brymbo	LL115FD
2	Marford	LL128TX
3	Borras	LL139QW
4	Brymbo	LL116AB
5	Borras	LL139QW
6	Gresford	LL112BG
7	Marford	LL128SZ
8	Brymbo	LL115FD
9	Borras	LL112AB

100% Windows (CRLF) UTF-8

output - Notepad

File Edit Format View Help

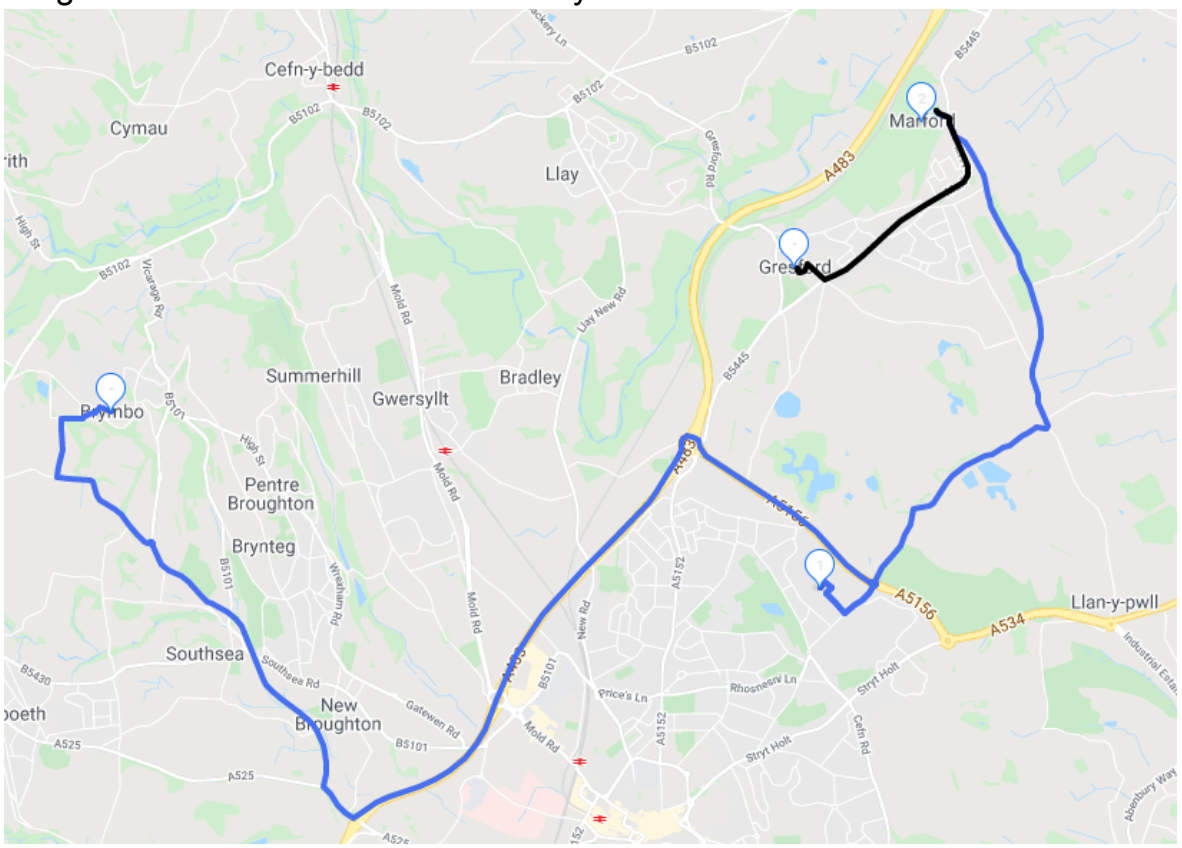
ID	AREA	POSTCODE
1	Brymbo	LL115FD
8	Brymbo	LL115FD
4	Brymbo	LL116AB
9	Borras	LL112AB
3	Borras	LL139QW
5	Borras	LL139QW
7	Marford	LL128SZ
2	Marford	LL128TX
6	Gresford	LL112BG

100% Windows (CRLF) UTF-8

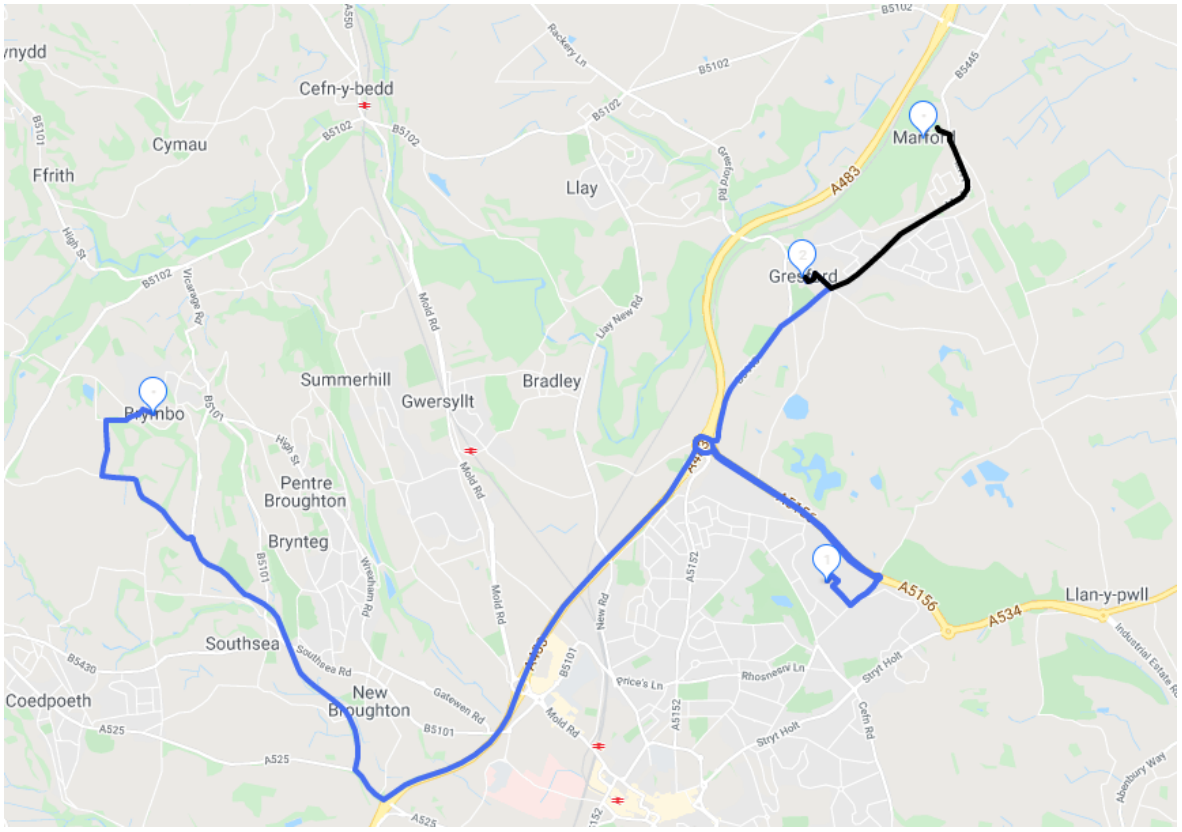
The above screenshot shows the data used by the **Timetable Calculations** class, and the result of it sorting. For viewing-convenience the data has

been put into plain text format. The data on the left has been sorted by area, and then sorted within each area by postcode, to create the order shown on the right.

However, the order on the right is not correct. To demonstrate this, the image below shows the route from Brymbo > Borras > Marford > Gresford.



As can be seen, this route is inefficient, as it is easier to go from Borras to Gresford, and then to Marford (as shown below)



Below is part of the code that handles this.

```
113         Dim distances = New String(3, 3) {{999, 13, 10, 9}, {14, 999, 15, 14}, {10, 15, 999, 5}, {7, 14, 5, 999}} 'HARD CODED TABLE OF DISTANCES BETWEEN AREAS
114         'search current area for lowest distance
115         Dim lowestValue = 999, y = 0
116         For x = 0 To 3
117 
118             If distances(counter, x) < lowestValue Then
119                 lowestValue = distances(counter, x)
120                 y = x
121             End If
122 
123             'remove current area from list, to prevent double-bookings
124             distances(counter, x) = 999
125             distances(x, counter) = 999
126 
127         Next
128         'locate name of nearest area
129         If lowestValue = 999 Then 'all areas have been nullified, meaning they are already sorted
130             MsgBox("Done")
131         Else
132             Dim nextArea = areas(y)
133             Sort(nextArea) 'recursion, sort nearest area
134         End If
```

The error is caused as the distances array is an array of string values storing numbers (line 113), whereas it should be an array of integers. Changing this allowed the system to perform the data comparisons correctly.

```
113         Dim distances = New Integer(3, 3) {{999, 13, 10, 9}, {14, 999, 15, 14}, {10, 15, 999, 5}, {7, 14, 5, 999}} 'HARD CODED TABLE OF DISTANCES BETWEEN AREAS
```

And so generate the outputs correctly:

input - Notepad

File Edit Format View Help

ID	AREA	POSTCODE
1	Brymbo	LL115FD
2	Marford	LL128TX
3	Borras	LL139QW
4	Brymbo	LL116AB
5	Borras	LL139QW
6	Gresford	LL112BG
7	Marford	LL128SZ
8	Brymbo	LL115FD
9	Borras	LL112AB

100% Windows (CRLF) UTF-8

*output - Notepad

File Edit Format View Help

ID	AREA	POSTCODE
1	Brymbo	LL115FD
8	Brymbo	LL115FD
4	Brymbo	LL116AB
9	Borras	LL112AB
3	Borras	LL139QW
5	Borras	LL139QW
6	Gresford	LL112BG
7	Marford	LL128SZ
2	Marford	LL128TX

100% Windows (CRLF) UTF-8

16/12/202	Timetable data being stored incorrectly

bookings - Notepad

File Edit Format View Help

ID	cID	jID	Len	Day	Start8	2	20	1	0
4	3		45	1	20				
9	7		60	1	65				
3	6		15	1	125				
5	9		120	1	140				
6	5		480	2	0				
7	8		10	2	480				
2	4		120	3	0				

Ln 1, Col 1 100% Windows (CRLF) UTF-8

The above screenshot shows that the first booking (ID 8 in this case) is being stored on the header line, instead of underneath it. Beneath is the code responsible for writing the data to the file.

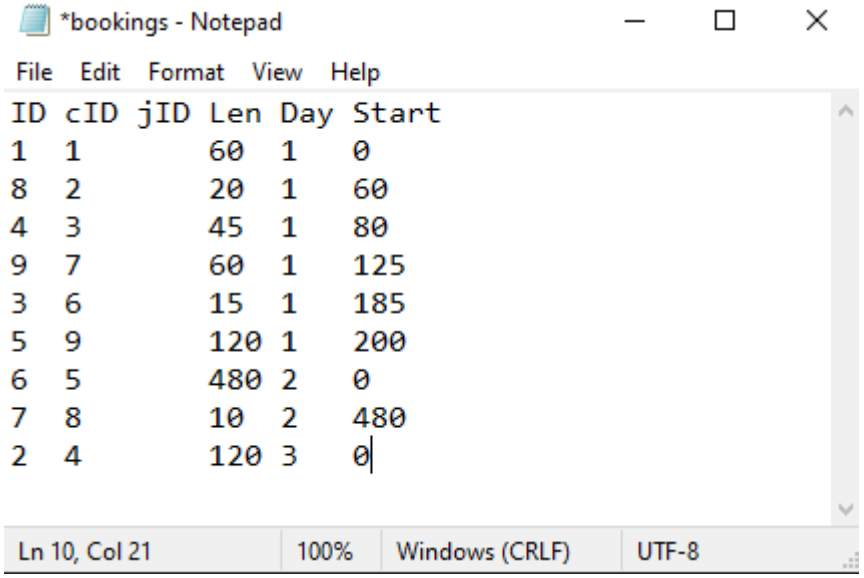
```
46      'store back into file
47      Dim tempBooking As FileHandling.booking
48
49      bookingsFileHandler.ClearFile() 'wipe file
50
51      For x = 0 To BookingList.Count - 1
52
53          tempBooking.ID = LSet(BookingList(x).ID, 3)
54          tempBooking.customerID = LSet(BookingList(x).customerID, 4)
55          tempBooking.jobID = LSet(BookingList(x).jobID, 4)
56          tempBooking.jobLength = LSet(BookingList(x).jobLength, 4)
57          tempBooking.day = LSet(BookingList(x).day, 4)
58          tempBooking.startTime = BookingList(x).startTime
59
60          bookingsFileHandler.WriteLineToFile(tempBooking.ID & tempBooking.customerID & tempBooking.jobID & tempBooking.jobLength & tempBooking.day & tempBooking.startTime)
61      Next
```

```
72      Sub ClearFile()
73
74          System.IO.File.WriteAllText(Application.StartupPath & "\" & fileName, fileHeader)
75
76      End Sub
```

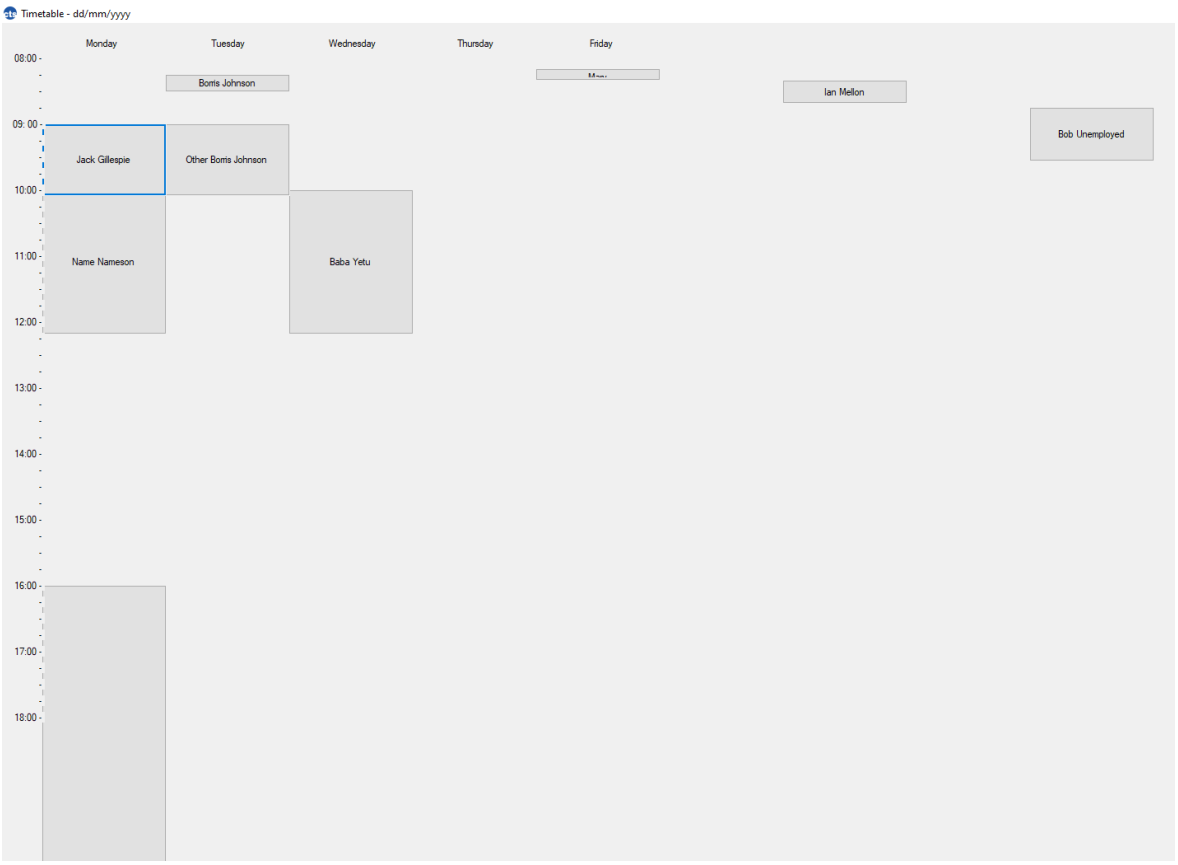
This issue is caused as wiping the file (which happens every time a new timetable is generated) writes to the line, and then when writing new lines to the file, it adds to the current line, instead of writing to a new line. This was easily amended by writing a new line immediately after wiping.

```
72      Sub ClearFile()
73
74          System.IO.File.WriteAllText(Application.StartupPath & "\" & fileName, fileHeader)
75
76      End Sub
```

Which resulted in the data now being correctly written to the file.



17/12/2020	**Timetable places items on incorrect days**



This screenshot shows the error caused when trying to view the timetable. The items are only meant to populate Monday, Tuesday and Wednesday, and should not go beyond 4pm.

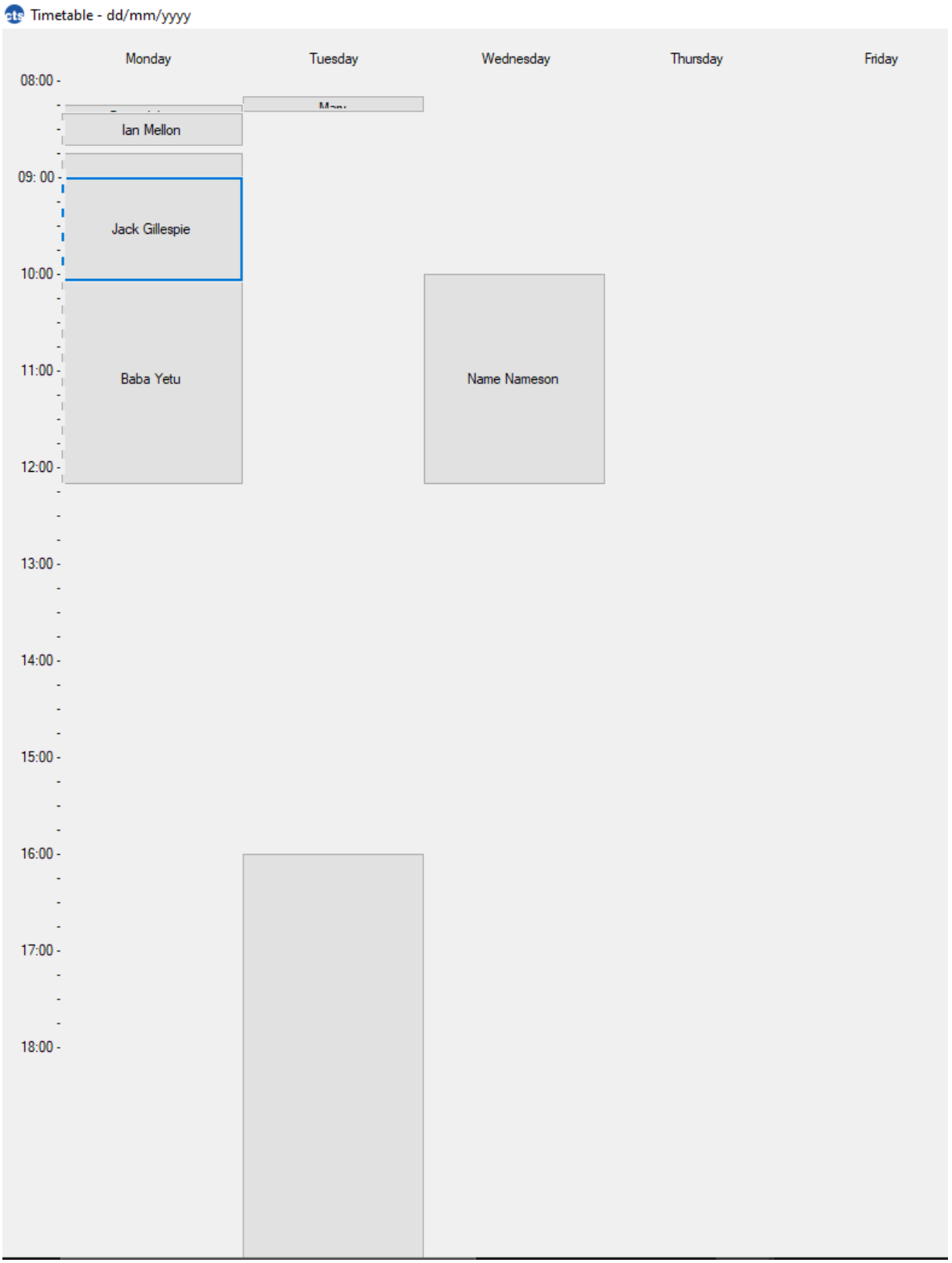
This image however shows items on Friday, and to the right, off of the timetable. As well as this, the last item on Monday exceeds the 4pm time limit.

Below is the code that generates the timetable.

```
42          'RUN THROUGH BOOKINGS, CALCULATE SIZE AND CO-ORDS OF BUTTON, INSTANTIATE BUTTON
43          For x = 0 To bookingsList.Count - 1
44
45              Dim button = New Button()
46              Dim buttonSize As Size, buttonLocation As Point
47
48              Dim jobLength = Convert.ToInt16(bookingsList(x).jobLength)
49              Dim startTime = Convert.ToInt16(bookingsList(x).startTime)
50              Dim day = Convert.ToInt16(bookingsList(x).day)
51
52              buttonSize.Width = 152
53              buttonSize.Height = 22 * (jobLength / 15)
54
55              buttonLocation.X = 48 + (startTime * 150)
56              buttonLocation.Y = 42 + (20 * (jobLength / 15))
57
58              button.Text = customersList(x).Name
59              button.Size = buttonSize
60              button.Location = buttonLocation
61
62              Controls.Add(button)
63
64          Next
```

The x axis of this timetable is the days of the week (as shown in the first screenshot). However, when the program calculates the x coordinates of the buttons (on line 55 of the above code), it calculates it by multiplying the startTime, whereas this should be the day of the week. Changing this (as seen below) resolved some of the issues.

```
55              buttonLocation.X = 48 + (day * 150)
```



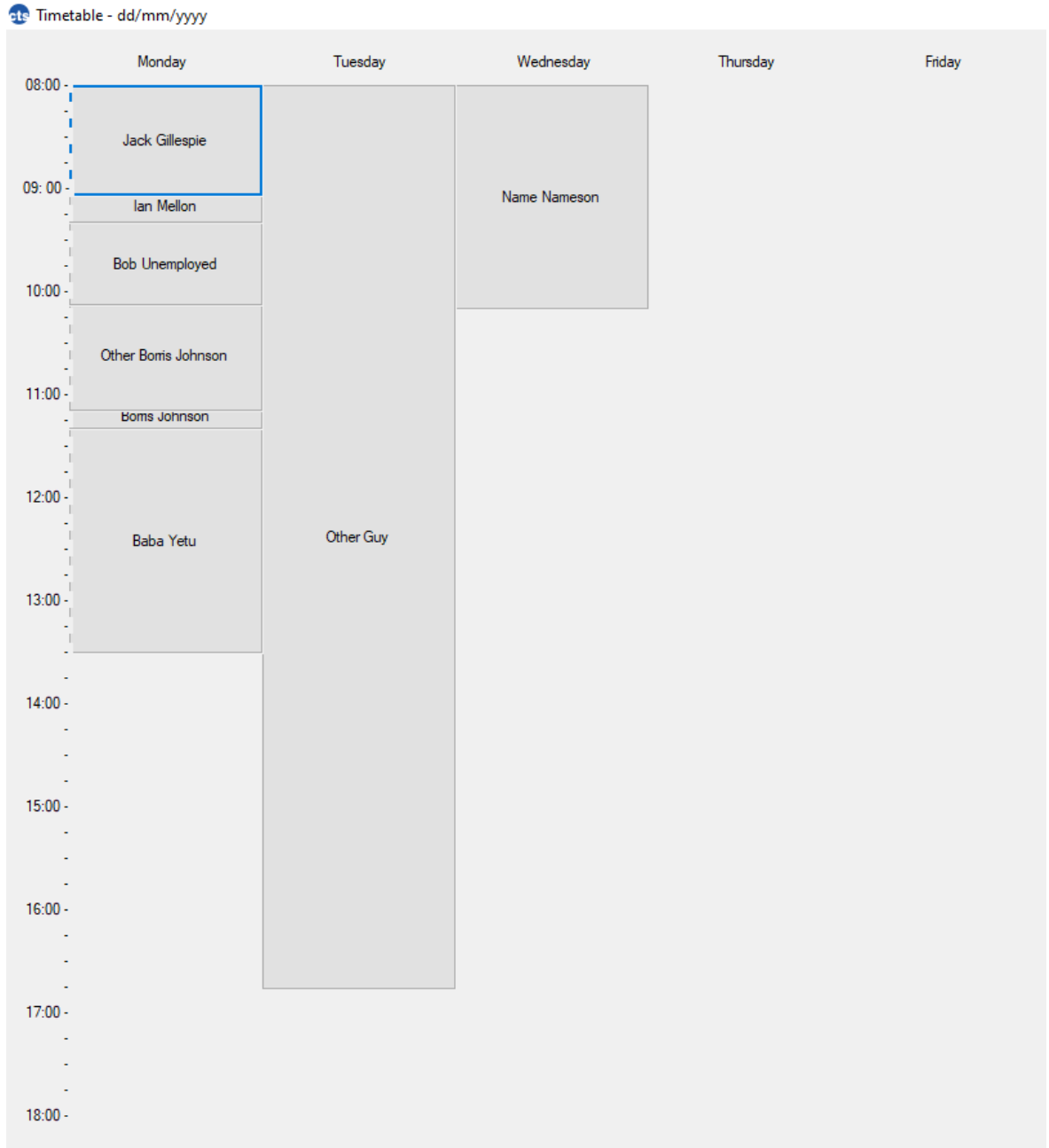
This has corrected the days on which the items are one, with the items being correctly aligned to Monday, Tuesday and Wednesday. However, the items are still incorrect along the y axis, by exceeding the 4pm limit, and by overlapping with each other.

The calculations for the y coordinates are on line 56, and calculate the

position using the jobLength. However, the coordinates are the starting point for the button, and the length of the job is the item's length along the y axis, and so using the length to determine the starting position is incorrect. To resolve this, jobLength on line 56 was replaced with startTime.

```
56          buttonLocation.Y = 42 + (20 * (startTime / 15))
```

This resulted in the image below, which shows all of the items correctly aligned, and not exceeding the 4pm or Friday limits.



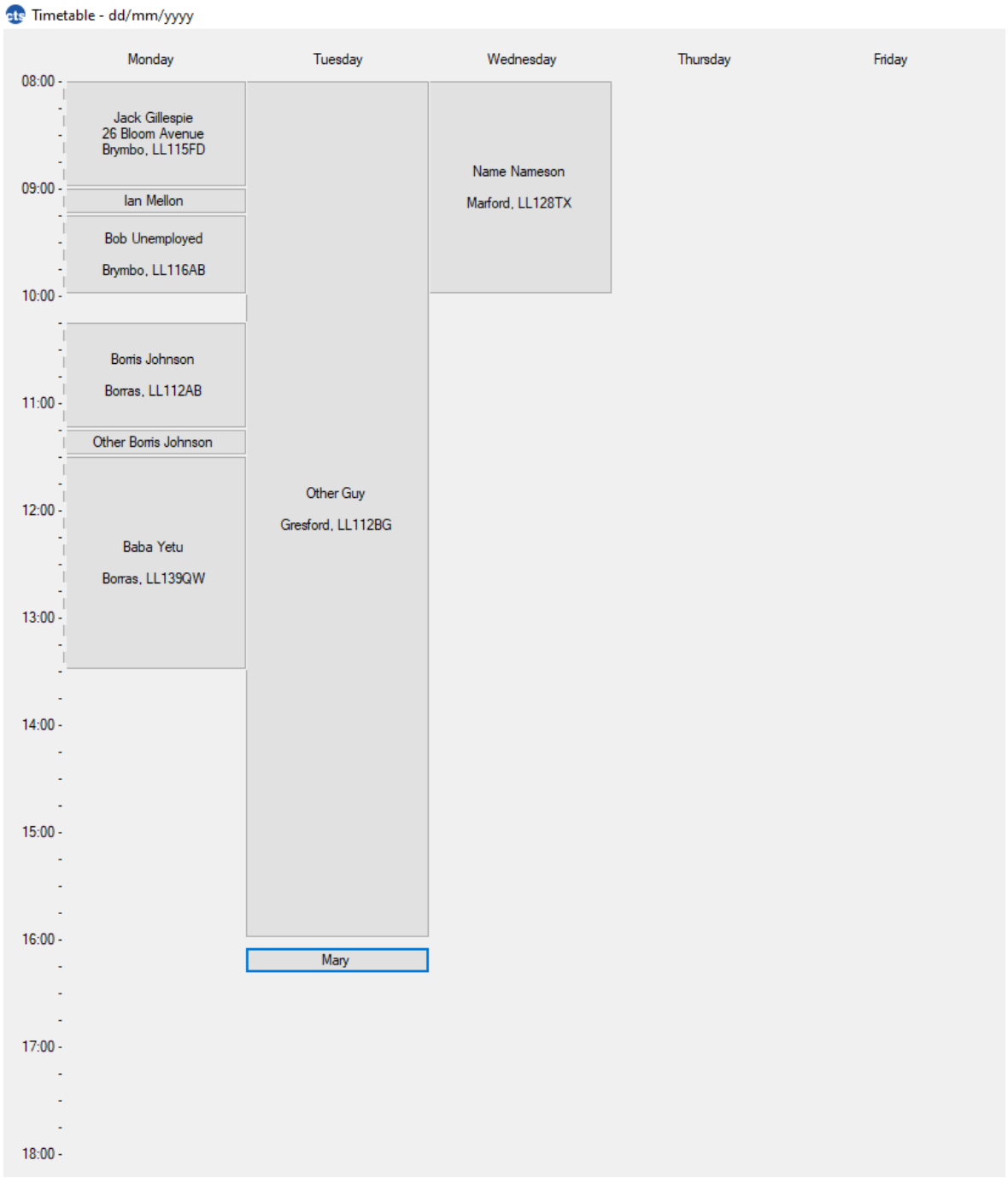
17/12/2020	Timetable spacing

The screenshot above, whilst now correctly aligned, is still slightly incorrect, as all of the items are back-to-back. Whilst not currently shown on the table, Jack Gillespie, Ian Mellon and Bob Unemployed all live in Brymbo, whereas

Borris Johnson, Other Borris Johnson and Baba Yetu all live in Borras. This timetable expects the user to go immediately from Bob Unemployed (Brymbo) to Other Borris Johnson (Borras), whereas in reality Borras is a 15 minute drive from Brymbo. To amend this,the timetable needs to have spaces between certain items to compensate for drive time. The code below handles the generation of the timetable.

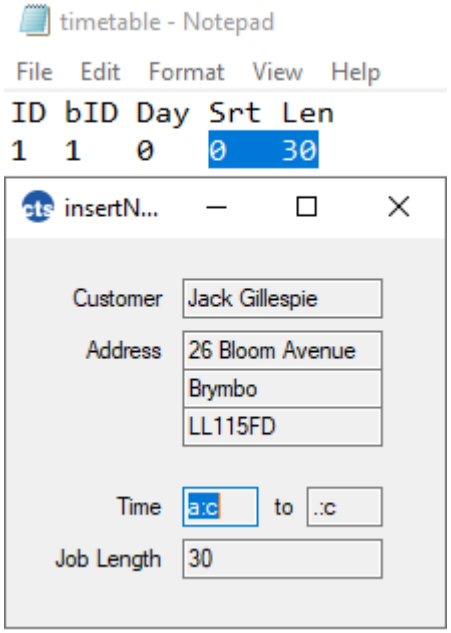
```
158         For x = 0 To NewBookingList.Count - 1
159             jobLength = Convert.ToInt16(NewBookingList(x).jobLength) 'required for calculations
160 
161             'if change of area occurs
162             If NewBookingList(x).area = currentArea Then
163             Else
164 
165                 'find distance between old and new areas
166                 Dim newArea = NewBookingList(x).area
167                 Dim distancesX As Integer, distancesY As Integer
168 
169                 For areasCounter = 0 To areas.Length - 1
170 
171                     If areas(areasCounter) = currentArea Then
172                         distancesX = areasCounter
173                     ElseIf areas(areasCounter) = newArea Then
174                         distancesY = areasCounter
175                     End If
176 
177                 Next
178 
179                 'add driving time to timetable
180                 Dim distance = distances(distancesX, distancesY)
181                 endTime += Math.Ceiling(distance / 5) * 5 'round UP to nearest 5, add to endtime
182                 currentArea = newArea
183             End If
184 
185             'fit booking into timetable
186             If endTime + jobLength > 600 Then 'if job does not fit into current day
187 
188                 day += 1
189                 endTime = 0
190 
191                 If day > 4 Then 'TEMP
192                     MsgBox("Error: Ran out of days.")
193                     Return
194                 End If
195 
196             End If
197 
198             startTime = endTime
199             endTime = startTime + jobLength
200 
201             currentBooking = NewBookingList(x)
202             currentBooking.day = day
203             currentBooking.startTime = startTime
204 
205             BookingList.Add(currentBooking) 'loads the now sorted and timestamped items back into the original list
206 
207         Next
```

By adding lines 179 to 182 into the code, the program will add a blank buffer in between items when a change of area is detected, by adding the drive time to the endTime variable. The result of this addition is the screenshot below:



As shown in this screenshot, there are now gaps between Brymbo to Borras, and Gresford to Marford. Whilst slightly hard to see, the first buffer is 15 minutes whereas the second buffer is a smaller 5 minute gap, as the distance between Gresford and Marford is very small.

19/12/2020	**Error viewing times of timetable items**



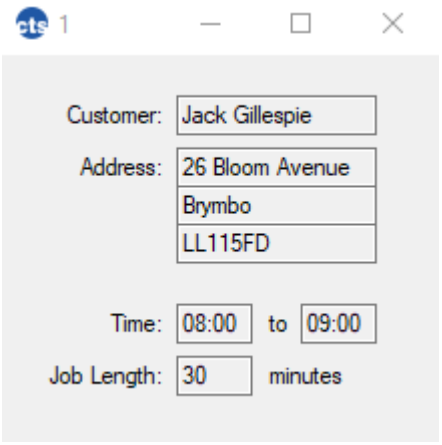
The above image shows that the timetable is displaying a start time of a:c, whereas it should be 08:00, and an endtime of .:c, whereas it should be 08:30. The code that generates these values are shown below.

```
15      Dim hour As String = ToString(Math.Floor(startTime / 60) + 8)
16      Dim minute As String = ToString(startTime - (Math.Floor(startTime / 60) * 60))
17      startTimeBox.Text = hour + ":" + minute
18
19      hour = ToString(Math.Floor((endTime + jobLength) / 60) + 8)
20      minute = ToString((endTime + jobLength) - (Math.Floor((endTime + jobLength) / 60) * 60))
21      endTimeBox.Text = hour + ":" + minute
```

This error is caused as the code is converting the numerical values into strings so they can add the ‘:’ to them, however in doing so, the numerical values are being converted to their ASCII counterparts. To amend this, the following code was added below:

```
18      Dim hour As String = Math.Floor(startTime / 60) + 8
19      Dim minute As String = startTime - (Math.Floor(startTime / 60) * 60)
20      If hour.Count = 1 Then
21          hour = "0" + hour
22      End If
23      If minute.Count = 1 Then
24          minute = "0" + minute
25      End If
26      startTimeBox.Text = hour + ":" + minute
27
28      hour = Math.Floor((endTime + jobLength) / 60) + 8
29      minute = (endTime + jobLength) - (Math.Floor((endTime + jobLength) / 60) * 60)
30      If hour.Count = 1 Then
31          hour = "0" + hour
32      End If
33      If minute.Count = 1 Then
34          minute = "0" + minute
35      End If
36      endTimeBox.Text = hour + ":" + minute
```

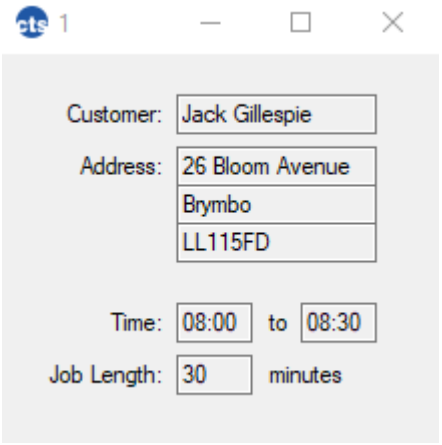
This code did fix the letters issue, as shown below, however, the end time was still incorrect.



By altering line 29 of the above code to ignore the jobLength, as the jobLength has already been used in line 28, and using it twice results in incorrect values.

```
28          hour = Math.Floor(endTime / 60) + 8
29          minute = endTime - (Math.Floor(endTime / 60) * 60)
30          If hour.Count = 1 Then
31              hour = "0" + hour
32          End If
33          If minute.Count = 1 Then
34              minute = "0" + minute
35          End If
36          endTimeBox.Text = hour + ":" + minute
```

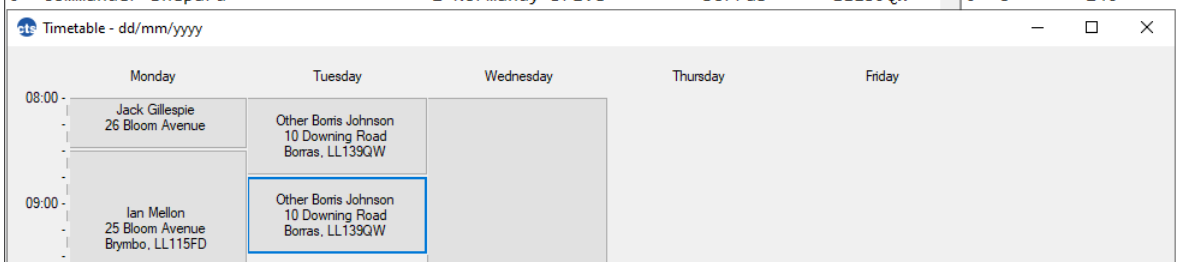
Doing this resulted in the corrected values being displayed.



20/12/2020	**Timetable duplicates items**

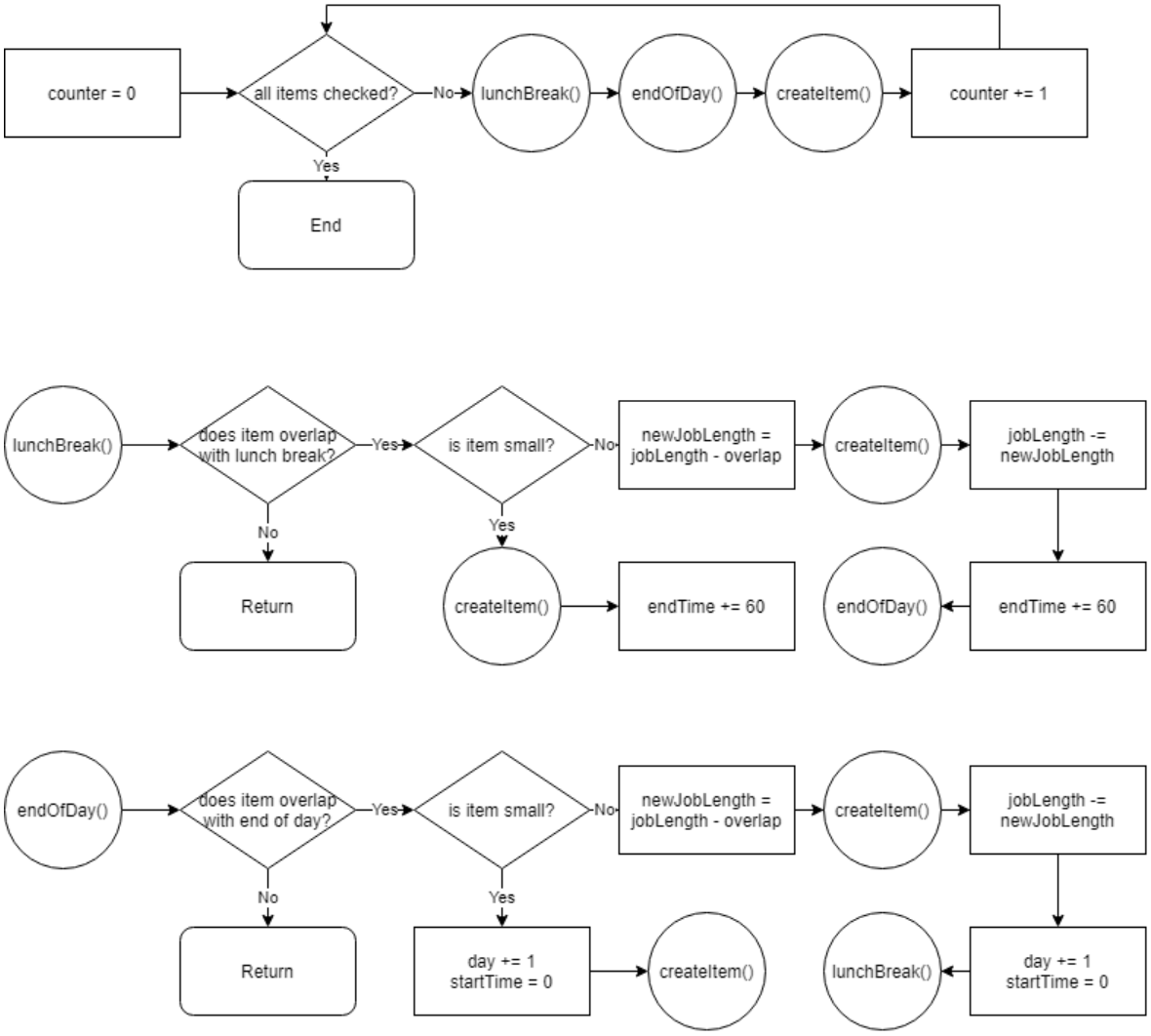
ID	NAME	PONUMBER	ADDRESS	AREA	POSTCODE
1	Jack Gillespie		26 Bloom Avenue	Brymbo	LL115FD
2	Ian Mellon		25 Bloom Avenue	Brymbo	LL115FD
3	Bob Unemployed		18 Ottowa Road	Brymbo	LL116AB
4	Name Nameson		15 Name Street	Marford	LL128TX
5	Other Guy		4 Meadows View	Gresford	LL112BG
6	Other Borris Johnson		10 Downing Road	Borras	LL139QW
7	Borris Johnson		10 Downing Lane	Borras	LL112AB
8	Mahatma Gandhi		The Taj Mahal	Marford	282001
9	Commander Shepard		2 Normandy Drive	Borras	LL139QW

ID	cID	jID	Len
1	1		30
2	3		45
3	5		15
4	7		30
5	9		60
6	2		90
7	4		330
8	6		45
9	8		240



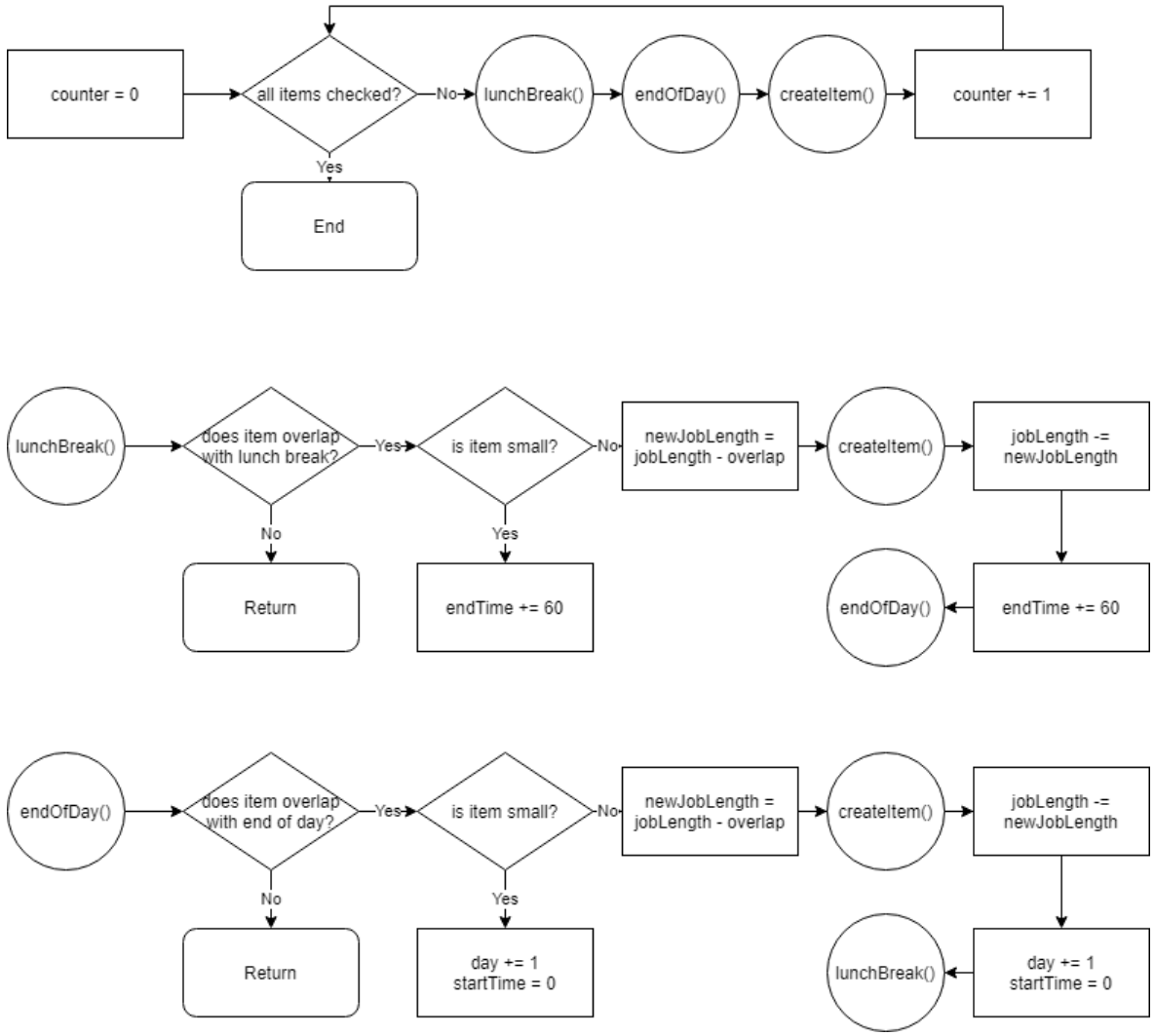
The above screenshot shows that Other Borris Johnson has been displayed on the timetable twice, despite the fact that his customer ID (6) is only in the bookings (top right) once, and not twice.

The code that handles the timetable generation is very long and quite complex, and so below is the flowchart that represents the process.

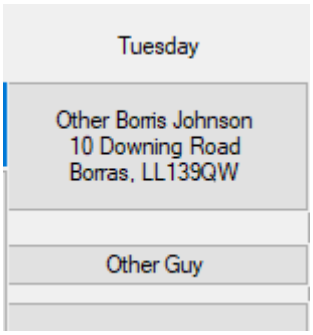


This process is the part of the program that adds lunch breaks and prevents items from exceeding the 4pm limit. As shown in this image, whenever an item overlaps with one of these limits, it will either create the item and alter the times, or alter the times and then create the issue. This means the item will be created regardless of which path it takes. However, the item is also created in the main loop (top of flowchart). This means that overlapping items will be created twice.

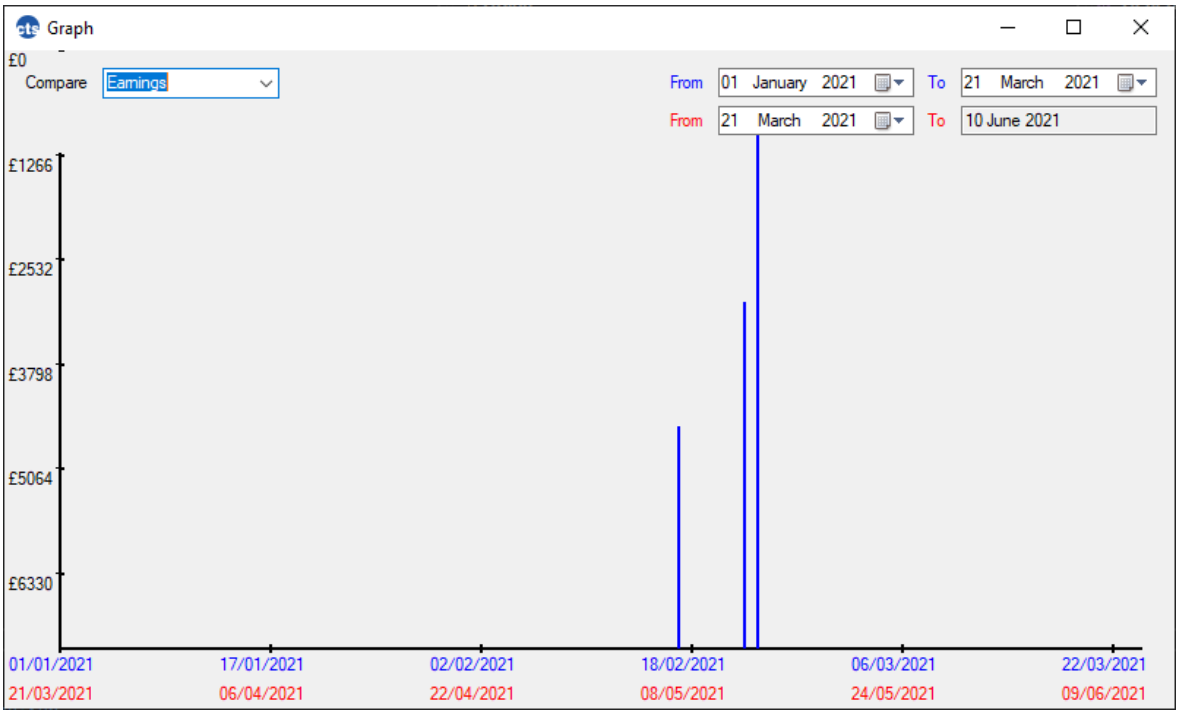
To amend this, the createItem() functions need to be removed from the 'Yes' branch of the 'is item small?' tree, like in the image below.



Doing this prevents the items from being created twice, and so result in the image below, with no duplications.



21/01/2020	Timetable y axis incorrect



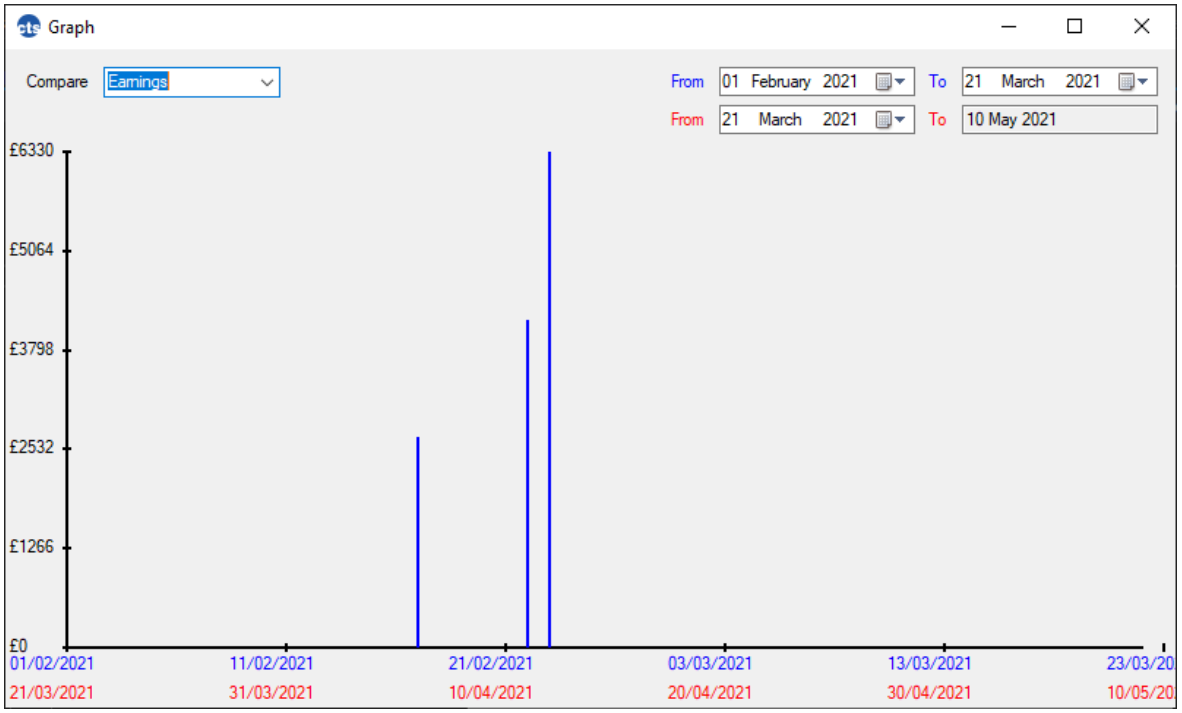
The image above shows a graph that has been generated. However, the y axis is inverted with £0 at the top, and not the bottom.

```
238         'label y axis
239         For y = 0 To 5
240             'maths
241             Dim multiplier As Integer = Math.Floor(maxValue / 5)
242             Dim value As Integer = y * multiplier
243
244             Dim newPosition As Point
245             newPosition.Y = (deltaY * value)
246             newPosition.X = 0
247
248             'CREATE LABEL
249             Dim newLabel As New Label
250
251             newLabel.Text = "£" & value
252             newLabel.ForeColor = Color.Black
253             newLabel.Location = newPosition
254             newLabel.Width = 37
255             newLabel.Height = 13
256             Controls.Add(newLabel)
257             labelsList.Add(newLabel)
258
259             surface.DrawLine(penBase, origin.X - 3, newPosition.Y, origin.X + 3, newPosition.Y) 'notch
260         Next
261
262     End Sub
```

The above code is responsible for creating this graph's y axis. The y coordinate is calculated on line 245, by using deltaY (the amount each £ is worth on the graph) multiplied by the amount of £s in the current item. In a visual basic form, (0,0) is the top left corner, with increasing x moving to the right, and increasing y moving down. The graph however has the origin point on the bottom left, with greater values moving up the graph (decreasing in y value). To correct this, line 245 can be changed to calculate the position relative to our origin point.

```
238         'label y axis
239         For y = 0 To 5
240             'maths
241             Dim multiplier = (maxValue / 5)
242             Dim value As Integer = y * (maxValue / 5)
243
244             Dim newPosition As Point
245             newPosition.Y = origin.Y - (deltaY * value) - 8
246             newPosition.X = 0
247
248             'CREATE LABEL
249             Dim newLabel As New Label
250
251             newLabel.Text = "£" & y * (maxValue / 5)
252             newLabel.ForeColor = Color.Black
253             newLabel.Location = newPosition
254             newLabel.Width = 37
255             newLabel.Height = 13
256             Controls.Add(newLabel)
257             labelsList.Add(newLabel)
258
259             surface.DrawLine(penBase, origin.X - 3, newPosition.Y + 8, origin.X + 3, newPosition.Y + 8) 'notch
260         Next
261
262     End Sub
```

Changing this results in the graph being correct along the y axis, as shown below.

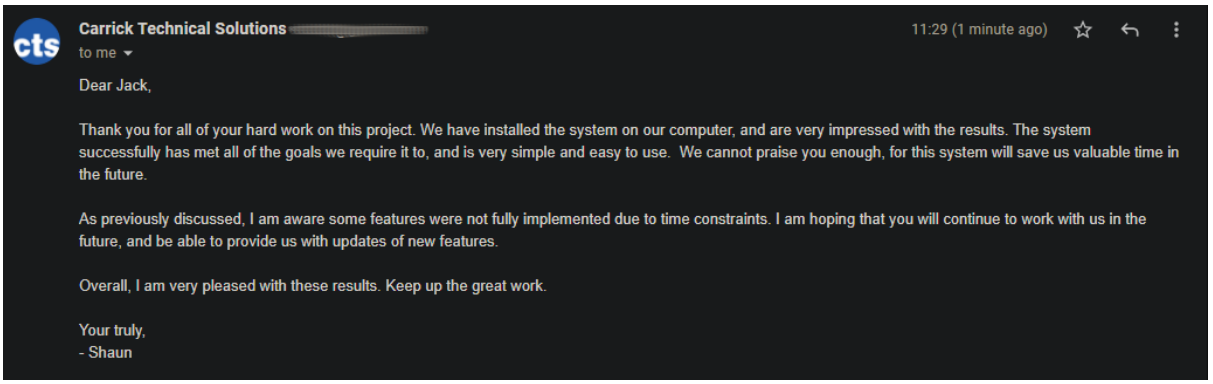


(The values have changed. This however is not due to the changed code, but is because different dates have been selected. This change does not affect the problem being tested here).

Evaluation

Feedback

I received the following email response from the director after deploying the application onto their computer system.



Carrick Technical Solutions [redacted] 11:29 (1 minute ago)

to me

Dear Jack,

Thank you for all of your hard work on this project. We have installed the system on our computer, and are very impressed with the results. The system successfully has met all of the goals we require it to, and is very simple and easy to use. We cannot praise you enough, for this system will save us valuable time in the future.

As previously discussed, I am aware some features were not fully implemented due to time constraints. I am hoping that you will continue to work with us in the future, and be able to provide us with updates of new features.

Overall, I am very pleased with these results. Keep up the great work.

Your truly,
- Shaun

I took this as a sign that I had successfully created a viable solution that successfully met their requirements. There are however some shortcomings to the solution that I can and will learn from, and plan to amend in future versions of the application.

Programming Language and Development Environment

For this project, I decided to use the Visual Basic programming language, in the Microsoft Visual Studio development environment. I elected to use this development tool due to the fact that I have fairly extensive experience with it, as I have used it regularly in the past for other projects. I chose to use Visual Basic, despite having a limited knowledge of the language, as opposed to other languages I am more experienced with, such as Python or C#, as Visual Basic is highly specialised for use in forms and the creation of Windows applications. Visual basic also makes use of a relatively simple syntax, not requiring ; terminators in the same way languages such as C# do. Despite choosing a language I was slightly inexperienced with, I feel that I was not very handicapped by this choice, as Visual Basic is a very beginner-friendly language and so was easy to adapt to, and is very well-documented, and so between my experience with other languages, and the use of the official documentation, I was able to adapt to the new language with only a small amount of problems encountered due to this inexperience.

As previously stated, Visual Basic is a very beginner-friendly language, making use of a simple syntax, meaning the programmer is not required to learn a complex language. As well as this, Visual Basic is designed primarily for the creation of Windows applications, whereas other languages such as C# are multi-purpose and so less specialised for the purpose I require it for. As well as this, Visual Basic (through Visual Studio) makes use of a visual and interactive IDE for the graphics-side of the application, using a super user-friendly drag and drop interface, so simple even non-programmers can easily use it. This greatly aided me with the visual side of the application, as only a very small amount of the visual outputs (graphs and tables) were produced using code instead of this tool. Another advantage of this language is that it contains a lot of inbuilt functions that allow programmers to easily create graphs and other visual features using the code. This was beneficial as it allowed me to make use of these functions, instead of having to manually code each and every aspect of the program. The language also has automatic control anchoring, allowing forms to easily be resizable without breaking or becoming impractical, as it is specially designed for making projects similar to mine.

Visual Basic is an object-oriented programming language, which suited this project very well, as the use of encapsulated objects is prevalent throughout due to the use of forms and classes. Python, whilst also a very

beginner-friendly language, and also a very powerful language, is not object-oriented, and is better suited for procedural and algorithmic programming.

Visual Studio has many helpful features, such as Intellisense, which suggested possible keywords and would allow for auto completion of statements by simply pressing tab. Not only did this speed up the process of typing, as I did not need to write every individual character, but also it would recommend the keywords it deemed to be valid in the given context, allowing me to easily find keywords that I otherwise may have struggled to determine due to my inexperience with the language. Visual Studio also allows the deployment of breakpoints, which pause the program during runtime and allows the programmer to view the contents of variables during this pause. This is an excellent aid during debugging, as it allowed me to determine which parts of the code were being run (for example during nested if statements,it can be used to determine which 'path' has been taken) by seeing which breakpoint has been triggered. It also allowed me to check the contents of variables, and so when incorrect data was produced, I could check if this was due to the program using incorrect data to begin with, or, if the input data was all correct, then I could determine it was the process itself that was at fault. This also aided me a great deal throughout this project, as I used it to determine the source of the majority of non-fatal errors that occurred, and so was able to address them faster than having to debug through alternative methods, such as the use of msgbox statements, for example.

There are some shortcomings to this choice however. As I do not own the full version of Visual Studio, I was required to use the free personal version of it. Whilst this wasn't overly restricting, some features, such as viewing class diagrams were restricted, and so I was unable to make use of these. As well as this, I was required to use a slightly dated version of the IDE for some of the programming work that was done using the college's facilities, and so I was unable to install the latest versions. This meant that I was unable to use some newer features, and was missing some quality of life updates that likely would have benefitted me. These features could have benefitted me, for example the class diagrams would be useful as I could easily compare them to my UML and tree diagrams from my initial design, allowing me to see if I was following the design or if I was straying away from it too much. Whilst this would certainly be a useful feature, it is not essential, and so did not negatively impact the project too much.

Another downside of this choice is that Visual Basic is not the most efficient language when it comes to handling memory. Visual Basic is a GUI-based development tool with graphical aspects that require a significant amount of space. This means the program may struggle to run as the program becomes larger, and handles a significant amount of data. This is not a major issue, as the program is relatively simple and small, and only for a small company. However, should the client require the program to expand, or handle a significant amount of data, then this limitation may become more prevalent. Other languages, such as C++ have a much more efficient memory usage. This however is not a massive issue, but it may however cause issues in the future, should the program require significant evolution in size, for example, scaling to function for a significantly larger company. Once again though, this is not an overly significant issue, but is however worth noting.

Feature used	**What were they?**	**What did they do?**	**Why did this help?**
standard modules	File handling I/O	This allowed me to easily access and modify files, allowing for easy storage of data.	It took away the complexity of having to load the contents of the file into an array, as the pre-built functions handled this.
	toString	Allowed me to convert an integer to a string.	This allowed me to convert dates either selected from DateTimePicker boxes, or calculated, into strings that could be stored in files.
	DrawLine	Allowed me to create lines on the form.	Provided a simple method to create graphs, without having to use an extensive amount of code.
	DateTimePicker	Allowed me to easily implement user date-selection.	This allowed me to implement date-based features (such as the timetable) without

			having to extensively code to create it myself.
	DateTimePicker.Value.DayOfWeek	Allowed me to calculate a numerical value for the date of week with a single function.	This prevented me having to create code to perform complex algorithms such as Zeller's Rule or the Key Value Method to calculate the numerical values.
	Trim()	Allowed me to remove whitespace from strings.	Allowed me to easily interpolate the useful data from files by removing the whitespace padding used in the fixed length system, without having to run character-by-character comparisons.
Drag-and-drop		Allowed me to easily add, edit and resize controls on the form without having to use code.	This saved me time by not requiring me to create code-based solutions for the layout of forms. This was applicable to the majority of forms, and only adaptive forms (graphs and timetables) required coded solutions.
Intellisense		Allowed me to see recommendations based on what I was typing, and allowed me to apply them using tab.	This saved me time when typing as I was not required to type out every last character, and could instead auto -complete statements. The recommendations were also based off of the context of the already written code, and so would allow me

			to work out which features I needed by looking at the recommended functions.
Breakpoints		Allowed me to pause the program whilst running to view variables at any given point.	Allowed me to see which part of the code was running, as the breakpoints only trigger when the line they are deployed on is about to run. Being able to view variables' contents during the pause was very useful for debugging and determining the cause of errors.
Debug windows		Would give me detailed descriptions of where and why the code has resulted in an error.	This aided me when debugging as I could see what specifically had occurred (e.g stack overflow, attempted conversion of data type) and in which file and on which line the error occurred, allowing me to easily identify and solve the errors.

Other Example Solutions

The first solution I looked at during my Desk Based Research was a quote calculator by calconic.com. Calconic is a calculation service that offers a broad variety of calculators. They have had years of research and updates to improve their product, and so are very well polished, and such have a lot of strengths that I can learn from.

One thing I noted during my analysis was that the validation used by the system is very beneficial and useful, as it prevents the calculations being broken by any incorrect data that could be entered. Calconic performed this in real-time, and would prevent invalid data from even entering the text box. I

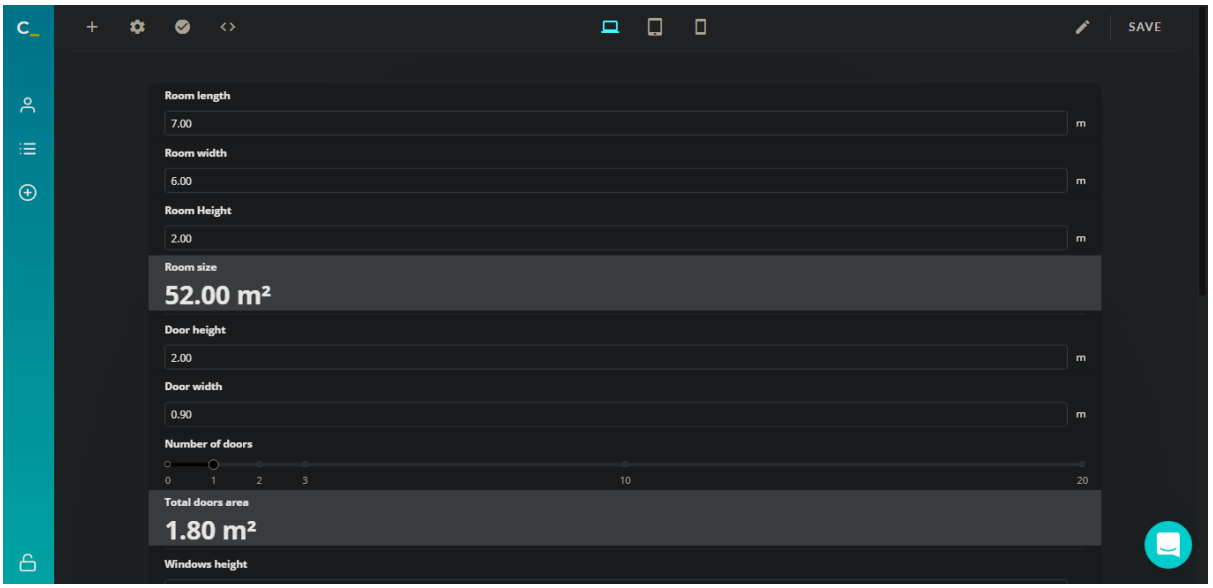
implemented a similar validation feature to my program, however, instead of having it run in real time, my validation system would only run when the user attempted to click on the ‘Add’ button. I did however, make the validation run when the user hovered over the button too. This decision was made to make the validation process happen sooner and highlight the errors to the user as quickly as possible, to save the user time, similarly to this feature.

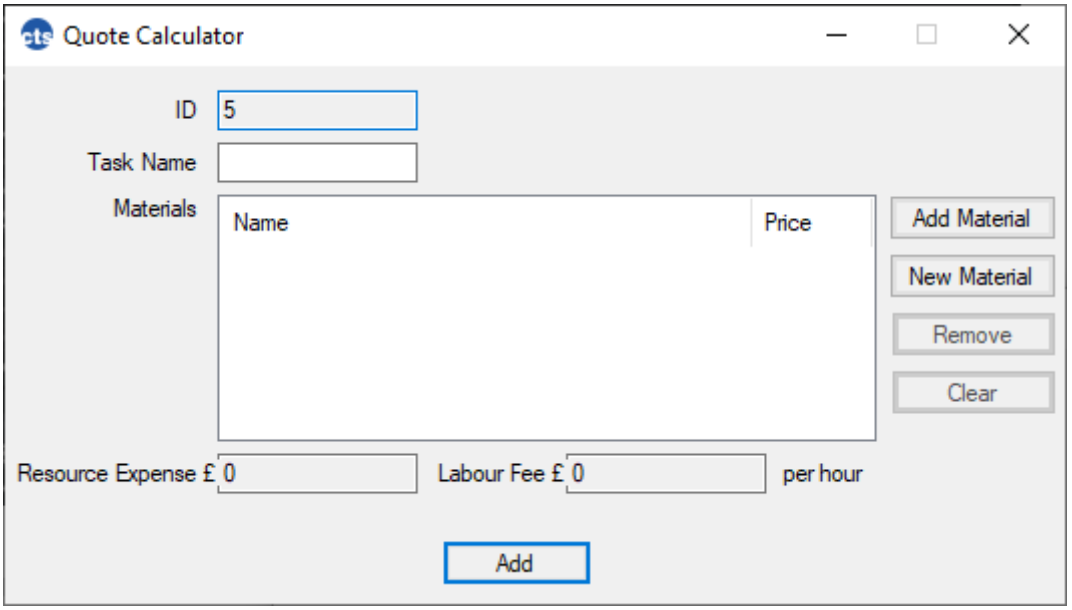
Another feature of this solution was the use of sliders (or dropdowns, which are similar) to prevent incorrect data being entered as the options are predefined with upper and lower limits. I replicated this solution by using ComboBox dropdown lists to restrict the options of input data where possible, as this reduces the opportunities the user has to enter invalid data that needs changing, once again, saving them time. I did not, however, implement any sliders into my solution, as very few of my numerical values would benefit from this. For example, monetary values can range from 0.01 to 999.99. This would be incredibly tedious to select the exact number required, as this range provides 99,998 different values which would be impractical, and annoying for the user.

Calconic used dropdown groups to prevent the user from being overwhelmed with an excessive amount of data input sections, as whole sections can be hidden when not in use. This is a useful feature, however as my data inputs were only ever a few boxes at a time, this would be more impractical than it would be useful, and so I did not implement this feature into my solution. The Calconic system also has the calculated data (such as the price) in bold, and larger than the regular input data so that it stands out to the user, making the system easier to use. I greyed out system-generated fields (such as the IDs), and set the cursor to turn into a stop symbol when hovered over, to visualise to the user that they do not need to alter these fields, and prevented the user from tampering with them.

An issue with the system was that it exclusively uses the metric system, whereas in this industry the imperial system is still very dominant, and some employees would prefer this option. In response to this, I planned to improve upon this flaw by allowing the user to change their input unit. However, as this system is only for a small amount of users, all of whom prefer the metric system (which I discovered upon asking them after performing this research) I did not make this option a priority. The users can use whichever system they want, as the names of the materials and jobs are user-defined (for example, ‘5m Length of Timbre’, and ‘Paint 5 Cubic Meter Room’, etc.). The system currently cannot convert these into the opposite system. However, as the

users all prefer the metric system it is not currently required, and can quite simply be added, should the company grow and the clients needs change.

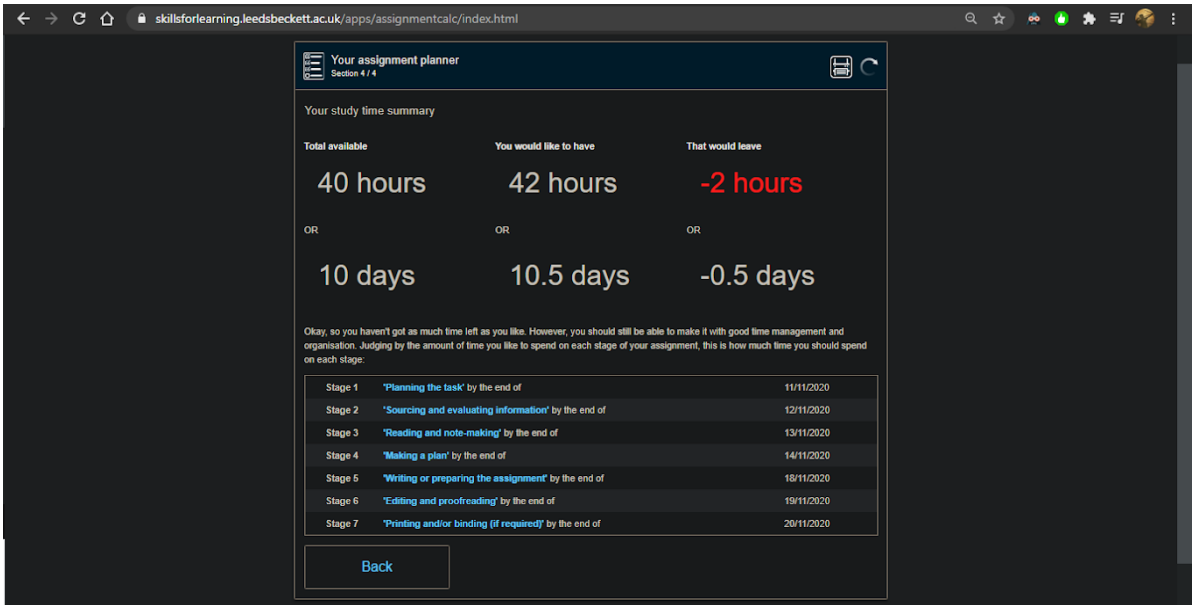


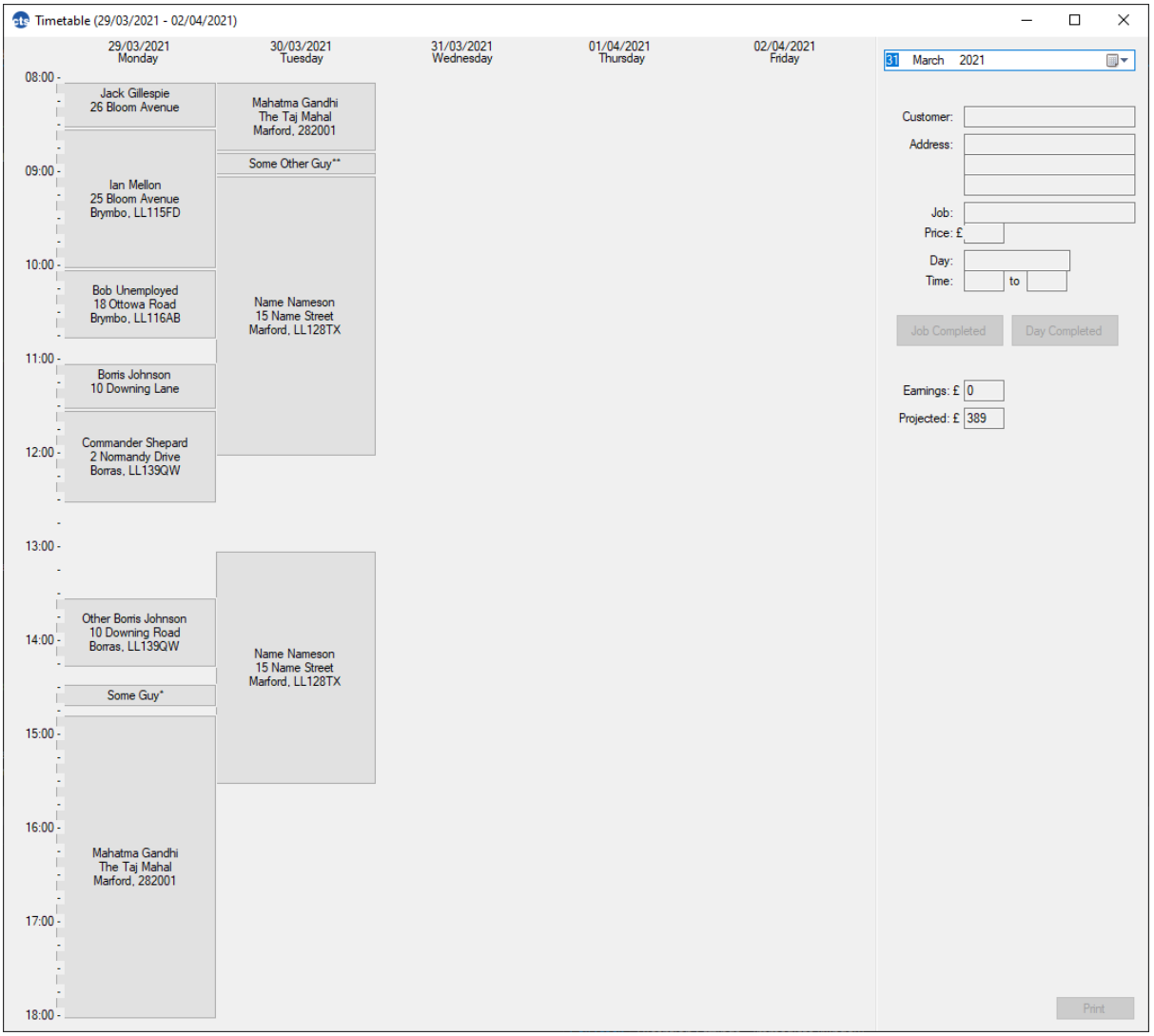


Another solution I used during my preliminary research was skillsforlearning. This system only books a single task, in order to help students manage their time for educational tasks and projects. However, the task is divided up into multiple subtasks, which act in the same way the bookings do in my system.

One feature I liked about this software was that it was very simple and easy to use. The calendar interface was very intuitive and prevented invalid dates from being entered. This system made this process as easy as possible, and handled all of the calculations. I attempted to replicate the simplicity of this solution, making use of DateTimePicker boxes to make date selection as simple and easy as possible. I also did my best to keep the number of fields to a minimum so not as to overwhelm the user.

A flaw I noticed with this software was the way that its data was output, as it outputs the data as plain text and was quite boring to read, as well as difficult for large quantities of data. I strived to learn from this mistake, and so created visual outputs where possible, such as the timetable and graphs. Below is their plaintext solution, and my more user-friendly visual solutions.





As shown here, my solution is easier when processing large amounts of data as the user is not overwhelmed with many lines of plain text.

The third system I investigated was a pathfinding software called RouteXL. This software is designed to calculate the shortest way of visiting several user-set stops.

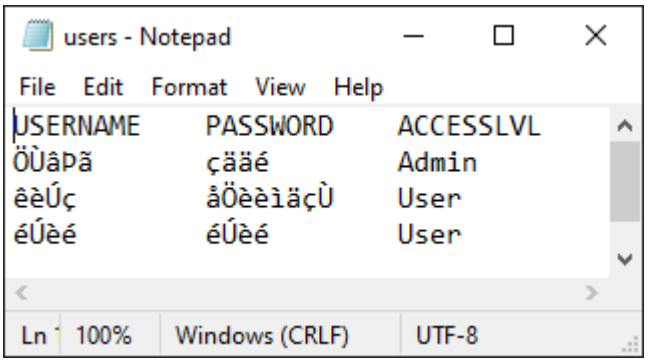
I noted that this system did an admirable job at making the process as easy as possible for the user. The system disallowed non-address data from being entered, meaning there is nothing that can break the calculations. I decided to implement format checks on fields to ensure that the user can only enter valid postcodes, and areas were selected from a dropdown list, as postcodes and areas are used in my timetable generation calculations, and so this prevents invalid data from affecting the calculations. Whilst the visualisation of the route was a very useful feature of this solution, I deemed it was too complex and not necessary for my system, and so elected to not try and replicate this feature.

It is worth noting that all of these solutions are professional and commercial solutions, likely created by a team of designers, developers and testers with financial backing. My solution, however, was a single-man solution with no financial backing, and under a strict time limit. This meant that the more complex parts of the solution were not viable, as I alone did not have the time and resources to spend on them. As I worked alone on this project, I was greatly limited by time, as I had to divide my time between multiple roles, for example I had to both create the system, and test it afterwards, whereas in professional development teams, programmers are allowed to focus solely on the development, whereas other team members focus on their own respective areas, such as the design and testing.

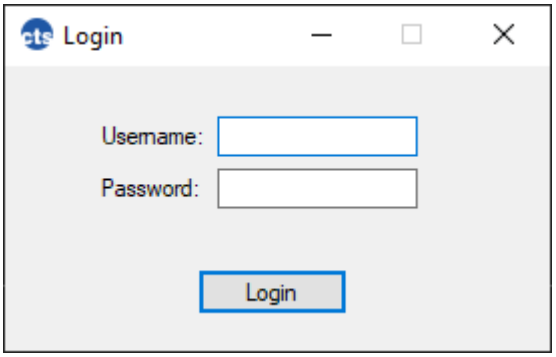
Evaluate Good Features and Shortcomings

Overall, I believe that this project was quite successful, with a simple and user friendly solution, that is able to effectively solve the complex task that the client needs.

Firstly, I believe the login system has been successful. This subsystem has successfully restricted access to the system to authorised personnel only, by preventing access to anyone without a valid username and password, but allowing access to the system if correct login details are input. This means complete strangers cannot access the system at all, making the system secure, and the access level system means that the system is secure from its own users too, as it prevents unauthorised users from accessing confidential parts of the system, preventing non-admin users from viewing the financial records of the system, and from editing and deleting information. This is significant, as if someone were to wipe the system it would be crippling to the company, as worse-case scenario all of the customer information would be lost, and best-case scenario, the company would have to re-digitise all of its paper records again. Either way this would be bad for the company, and so giving this ability only to trusted admins reduces the likelihood of this. This prevents customer data from being accessed by external parties, which is necessary in order to comply with GDPR regulations. This system makes use of encryption to prevent users from being able to work out usernames and passwords by accessing the users.txt file, as even if they do this, they will be shown an incomprehensible form of the login details. This successfully meets objective number 5 and 8, laid out in the investigation, '*All users must have securely encrypted usernames and passwords*.', and '*All staff should be able to access the required, and only the required, features and functions of the system*.'. This solution has successfully met and completed these requirements.



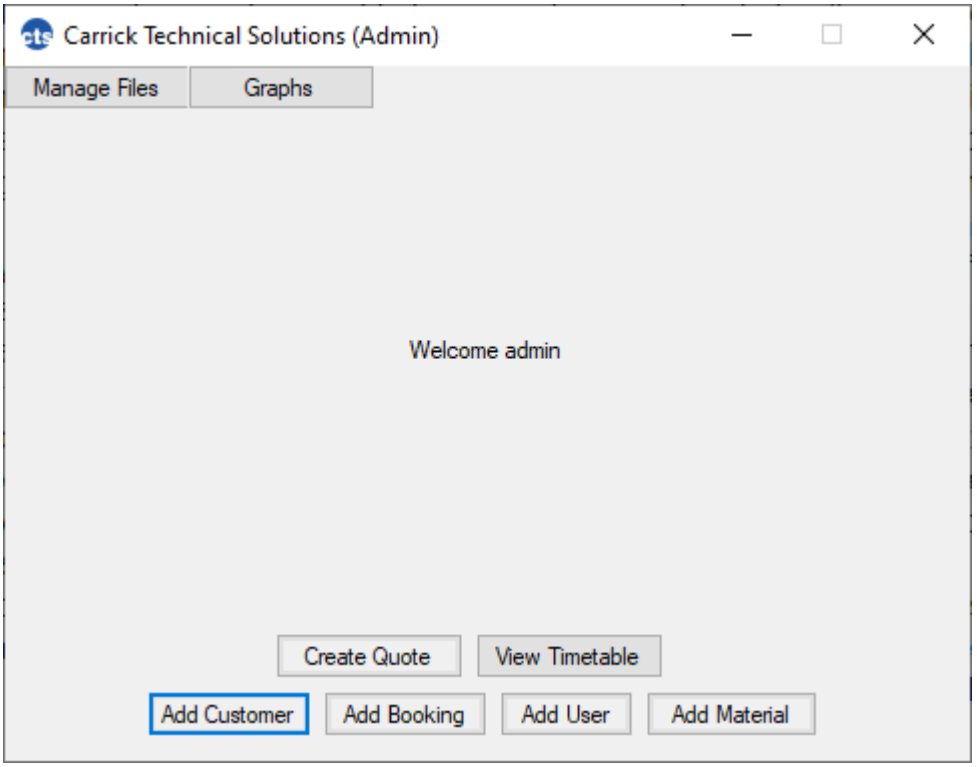
However, the encryption method is a Caesar cipher method, and could be replaced by a more complex hashing algorithm to further prevent the reverse-engineering of usernames and passwords. This would be even more secure. However, Carrick technical Solutions is a small company with 3 employees, and so very few people have access to the system this program will be installed on. The symmetrical encryption used should be enough to deter any unwanted-prying of details, as these people are trusted enough to not try and access confidential materials. If new features were added to the system, they could easily be set to be admin-only or available to all users, as the system used for this access level is modular and so allows for the addition of new components very easily. This system can quite easily be implemented into any other system, as it is self-contained, only using the encryption and file handling classes, and so is not dependent on the current system, and can be moved into another system, without causing too many complications. This function can be applied to almost any computer application, especially any that handles confidential information, or functions that can have negative effects (such as file erasion) that need restricting. This function can be improved by replacing the Caesar cipher with a more complex hashing algorithm if used in larger applications, or with more confidential data, to further protect it. When a new user is added using the Add USerform, the data is encrypted, fulfilling objective 22 '*New user data should be encrypted.*'.



The login screen makes use of text boxes and a button, allowing the input of data using a mouse and keyboard, as set out in the input objective 9: '*Username and password should be typed using a keyboard, and submitted using a mouse to*

interact with a button.'. This allows users to very easily interact with the form, without having to learn and adapt to new input methods, and instead using features they will already be familiar with from past experiences with computerised systems. This is important as I do not want there to be a steep learning curve for using this program, as it is designed to be as simple and easy to use as possible, and therefore by using familiar concepts where possible, should be easier for users to understand and learn.

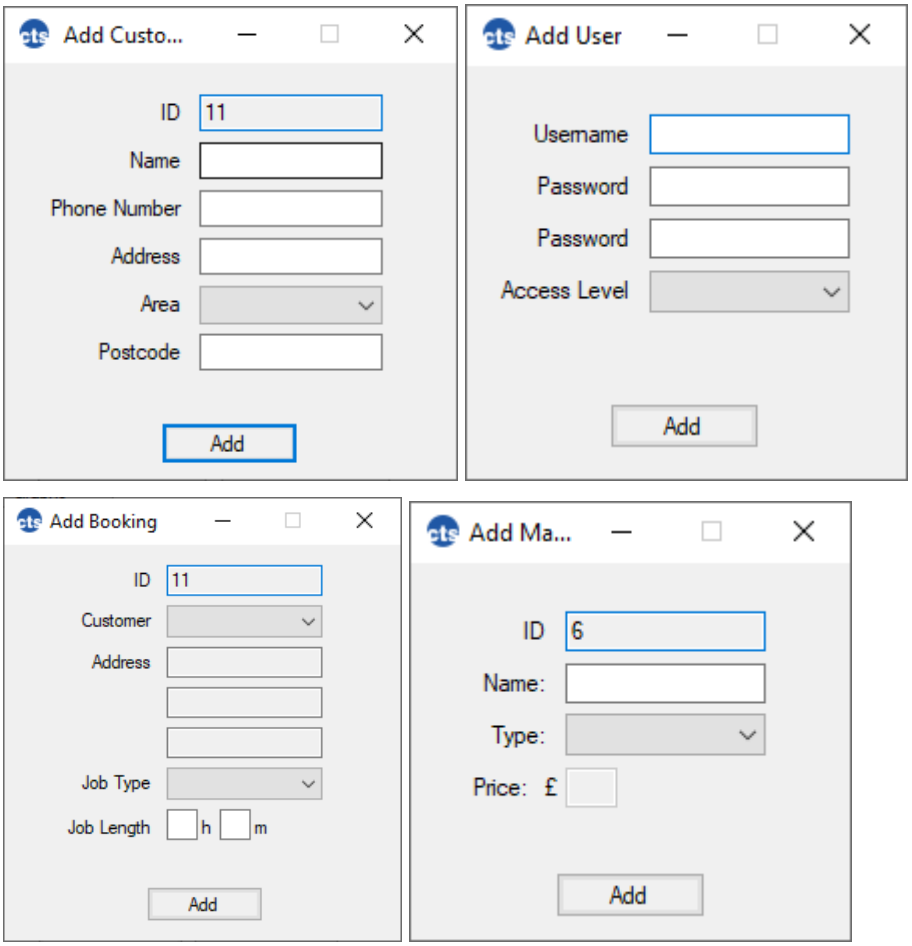
The central primary form is designed to be a simple navigation hub, allowing users to easily open the correct form that they require. This was designed to be as simple as possible and prevent being overly complex by having too many buttons and options. In the investigation objective 10 states '*The user should navigate to the required menu using the mouse to select the corresponding button.*'. By keeping this form simple and uncluttered, it makes it easier for users to quickly find the navigation control they are looking for and use it. This is a success from the design area of the project as a simple and intuitive design is beneficial and better than a messy and complex navigation.



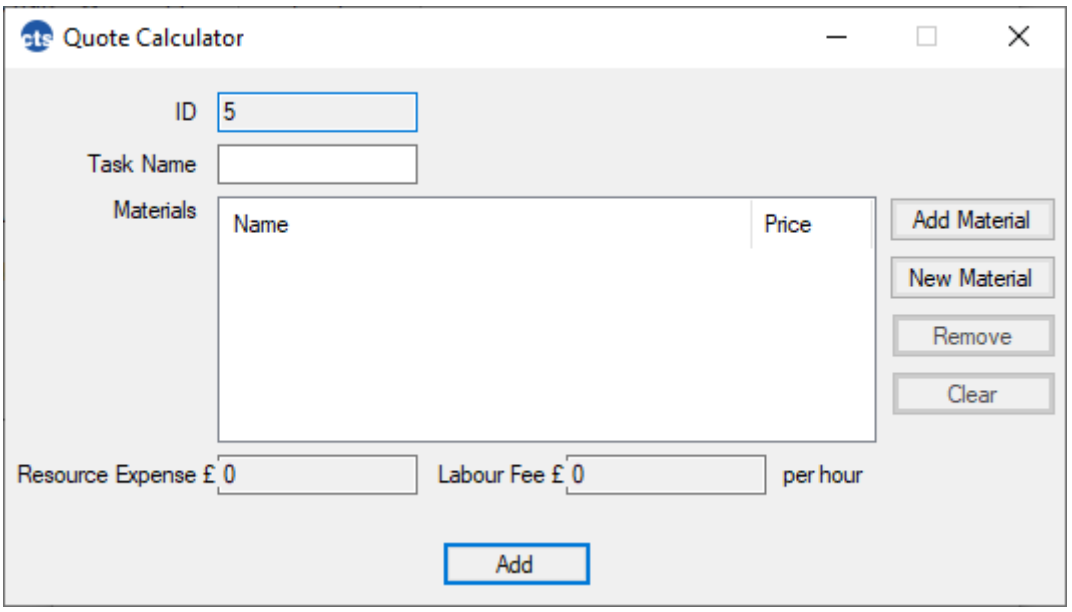
All primary forms are accessible directly from the main menu, this is slightly altered from the original design, as originally adding new customers, etc. was done from the file management form, however adding the option directly to the main form makes it easier if a user wants to just quickly add a record to a file without having to go through the longer process of using the file management form. Other forms, such as the Select Material form (used to add materials to add materials to a quote) are only necessary when other forms are in use (in this case, the Calculate Quote form) and so are only accessible from their respective parent forms, and not from the main

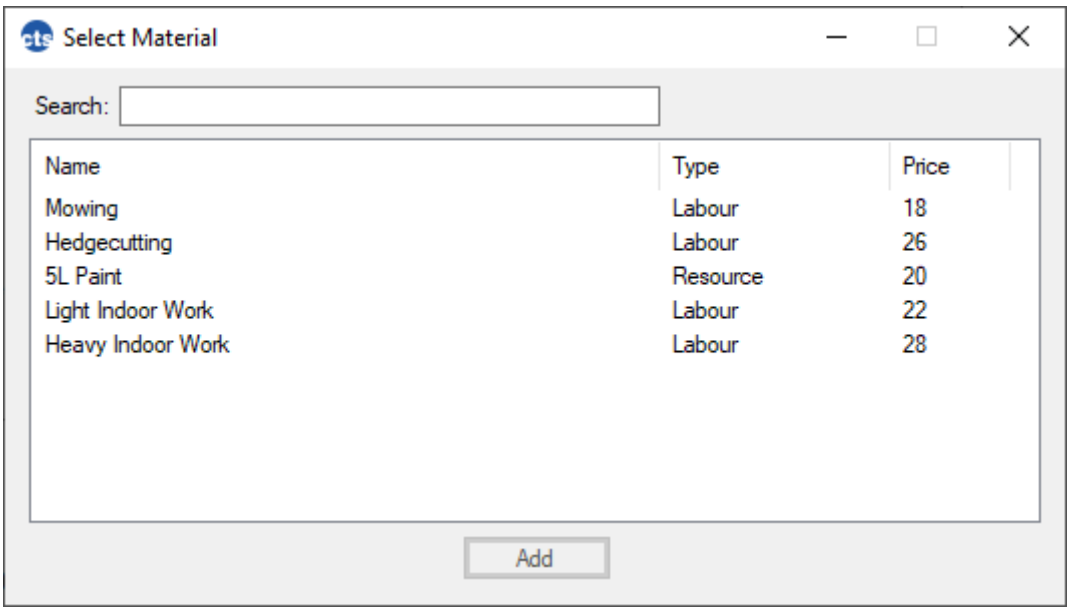
menu, as this would be an unnecessary addition that would only make the menu more cluttered and complicated. One upside to this approach is that new forms can easily be created, and new navigation buttons canesily be added to the menu to compensate for them, or, if more specific and less common forms are added, they can be easily added to their parent form.

The basic addition forms, such as Add User, Add Customer, Add Bookings and Add Material are all very similar, consisting of fields, labels (to identify the fields) and an add button. The goal of these forms was to allow the users to easily add entries into the files. Objective 7 is '*Users should be able to add entries and information easier.*'. In order to make this process as easy and simple as possible I used a minimal amount of fields to prevent the user from feeling overwhelmed. As well as this I made fields into dropdown boxes, to allow the user to simply select the item they want instead of having to type it, and risk mistyping an error. All of these design choices maximise the efficiency of the data addition forms, by reducing clutter and only requiring the user to manually type the minimum amount of information possible. This means the process will be faster than if the user had to manually type all of the data, which was a key part of the design: simplicity, ease of use, and time efficiency.



The Quote Calculator (also the Add Job form) are similar to the above forms, however this one is more complex, as it has a listview containing materials, which are added by selecting them from another form.



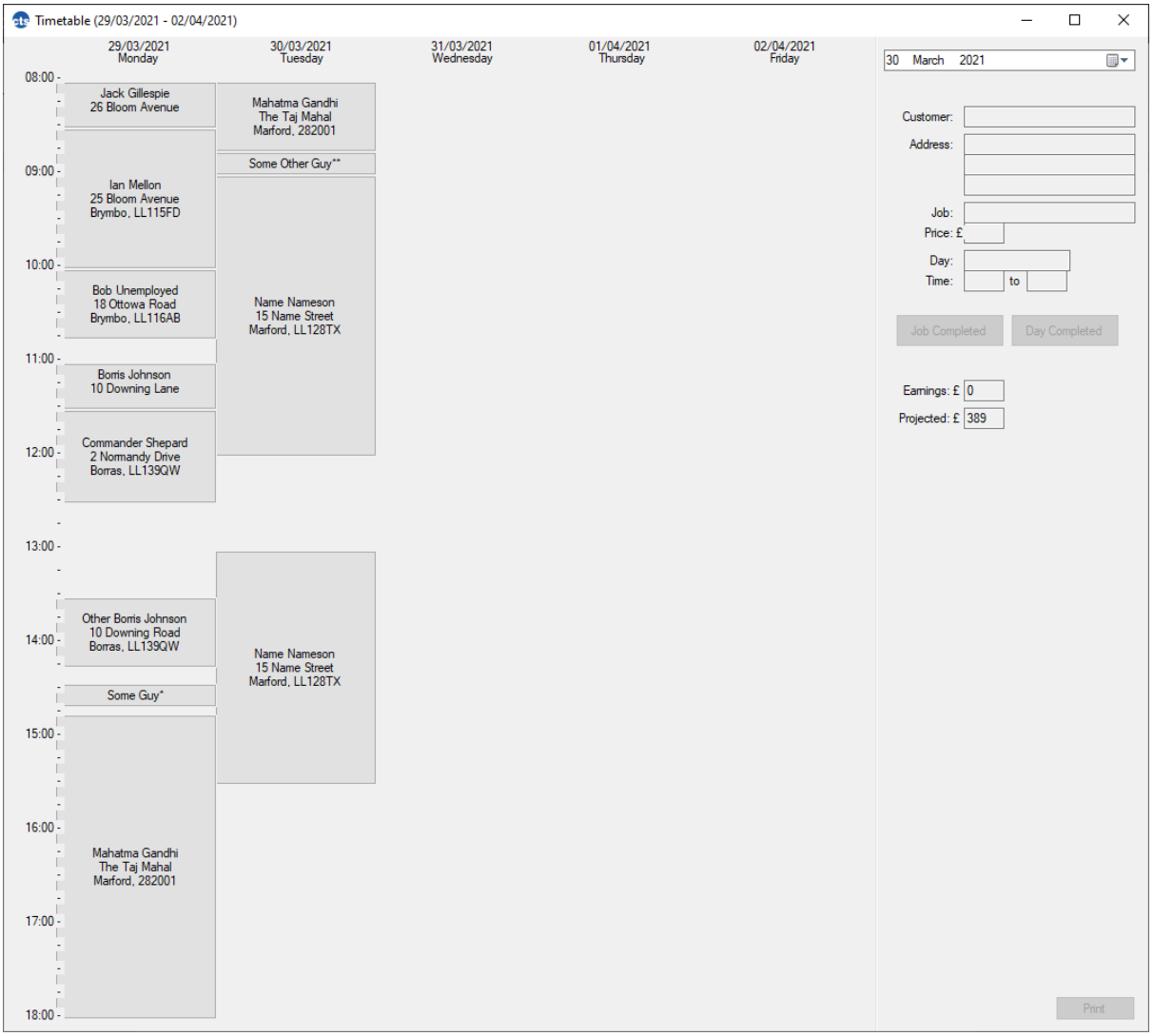


This design is very intuitive and so should be easy for people to understand and use. A search bar is available on the second form to allow the user to quickly find the item they want without having to scroll through what could become quite a large list. These features are all designed to create a simple and user-friendly system for inputting data into the system's files. After some basic testing, users were able to add data into the system up to twice as fast as they could using the old paper-based system.

The quote calculator automatically calculates the hourly rate of labour for the job, and the price of any resources required. As this is done by the machine

there is no risk of miscalculation due to human error when processing the numerical data. This is important as it will reduce the chance of customers being given incorrect quotes, and therefore are less likely to be given quotes thataretoocheap, resulting in the organisation losing money, or being given quotes are too high, possibly resulting in them taking their business elsewhere, as prices have to be competitive. These are both serious errors, which have happened in the past due to miscalculations, as I discovered during my investigation of the company from talking to Shaun, the director.

These forms also make use of validation checks such as presence checks, length checks, range checks, format checks, data type checks, unique checks and double-check verifications (explained extensively in the design and testing sections). This eliminates some of the possible errors when inputting data, removing the likelihood of invalid data being entered (though not preventing typos and incorrect information.). This covers objectives 14 - 19. This is significant as invalid data could be a common issue when entering data on a keyboard as opposed to writing on paper, especially for people who are less experienced at typing, and this data could prevent the system from working correctly. The major upside to this validation solution is that all of the validation checks are performed using an independent validation class, meaning that if a new input form is added in a future version, the validation class can be applied, ensuring the data is valid without having to create new code. This is important as it is the same code that has already been extensively tested, it can be implemented with dead certainty that it will function correctly for new forms. This class is also independent from the forms, and does not have any dependencies on them, and so could be taken and applied to any other Visual basic project,meaning it can be used for future projects and will be known to work.



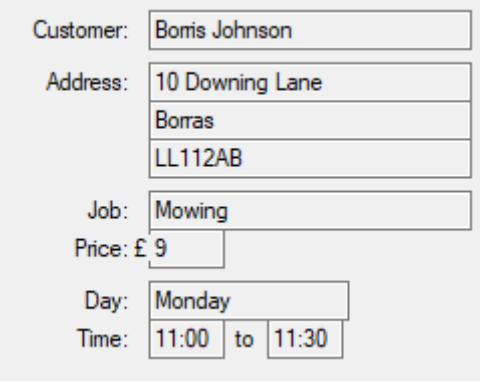
The timetable is a key part of the project. The system automatically creates this timetable, placing customers in the most efficient order, minimising the driving time between customers. The process also allows for an hour lunch break sometime around noon. This Completes Objective 20 '*Timetable should be generated to ensure that customers are ordered based on their geographical proximity.*'. The calculations for this are fairly complex, but are explained in-depth in the design section. This form displays the results of the timetable generation, in a simple and easy to understand visual table. This is beneficial as it prevents the user from trying to decipher a list of plaintext dates, and is instead an instantly recognisable and very common output method. This is significant as it can save the user valuable time, and items can easily be found due to each row being a new day, and the columns being a labelled time axis. If any part of this process was flawed and didn't work then this would create significant issues for the user, and so this part of the project has been extensively tested, to ensure that the correct results have been created, and that they are displaying correctly. This part of the project is

dependent on the data structure of the timetable items, however, can quite easily be adapted to display a timetable for any data, as long as a startTime, length and day are all available (or calculable). Creating this section was a very good learning opportunity, as it improved my geometric programming as I had to create algorithms for coordinate geometry calculations, which is a valuable skill for creating any visual data outputs.

Above is a part of the timetable screenshot, showing how it displays the projected earnings (money that will be earned if all jobs that week are completed). This was an objective of the project as it will allow for Shaun, the director, to have an idea of the week's income. This is significant as it allows Shaun to plan ahead for the company, as it gives him a strong indication of how much money the company will earn. For the present week, the confirmed earnings that have been earned so far will be displayed above the projected earnings.

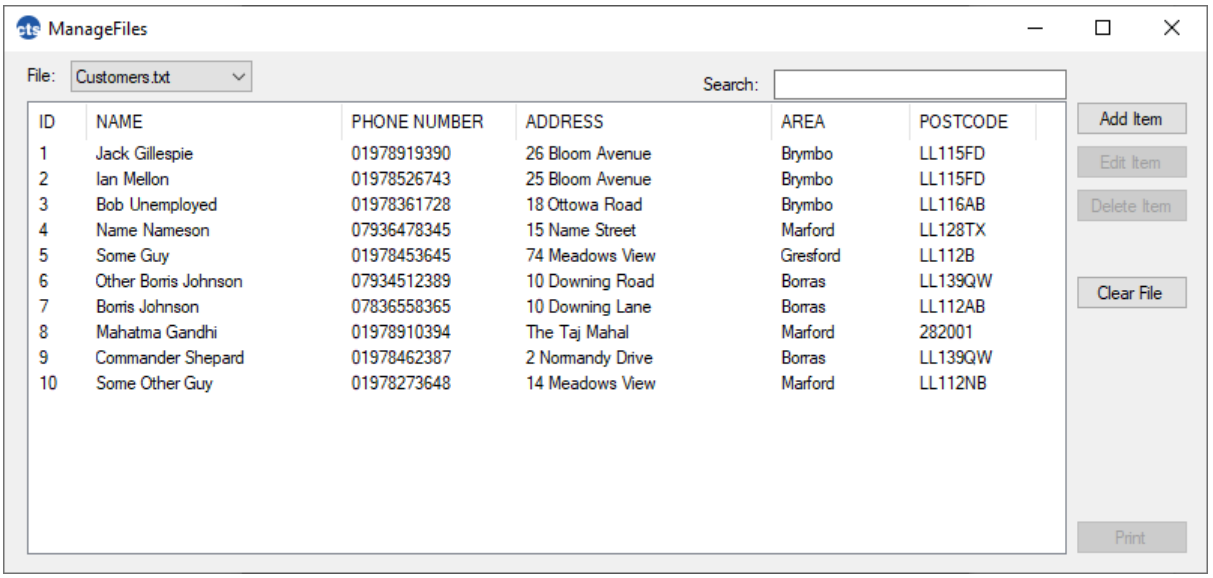
The timetable will also allow the users to view past timetable items that have been confirmed, for documentation and archival purposes. The program will display the total money earnt that week (but no projection, as the week has passed), as shown below.

This will be useful for the director as it will allow for him to view how much money he has earned and so can look at his official financial records to see if they differ, and if they do, he knows that he still needs to charge one of the customers, and so can use the list of names of the customers to see which one has been missed. The timetable will also allow the user to see all the information about the selected item, including the individual price of the job in question, and so the user can be sure what they need to charge and to whom.



These features are important for financial reasons as it allows the director to keep track of which jobs are done, and for how much without having to keep paper records (which are easily lost). As well as convenience, this system is also useful as it is significantly faster than generating the timetable mentally / manually on paper, and then having to write it down for future reference. When interviewing Shaun I was told that this process can take up around half an hour sometimes, whereas in testing, my system was able to generate the timetable in less than a few seconds. Not only was my system faster, saving the user valuable time, but it was a more efficient timetable, as it minimised the driving time between customers algorithmically, whereas the order of customers is worked out by the user using estimates when done manually. The impact of this is that this system can save the director hours of time over the course of the year by performing many of the calculations he has to perform for him.

My system also allows administrative users to edit, delete and even clear the contents of files. An image of the form that performs this is shown below.

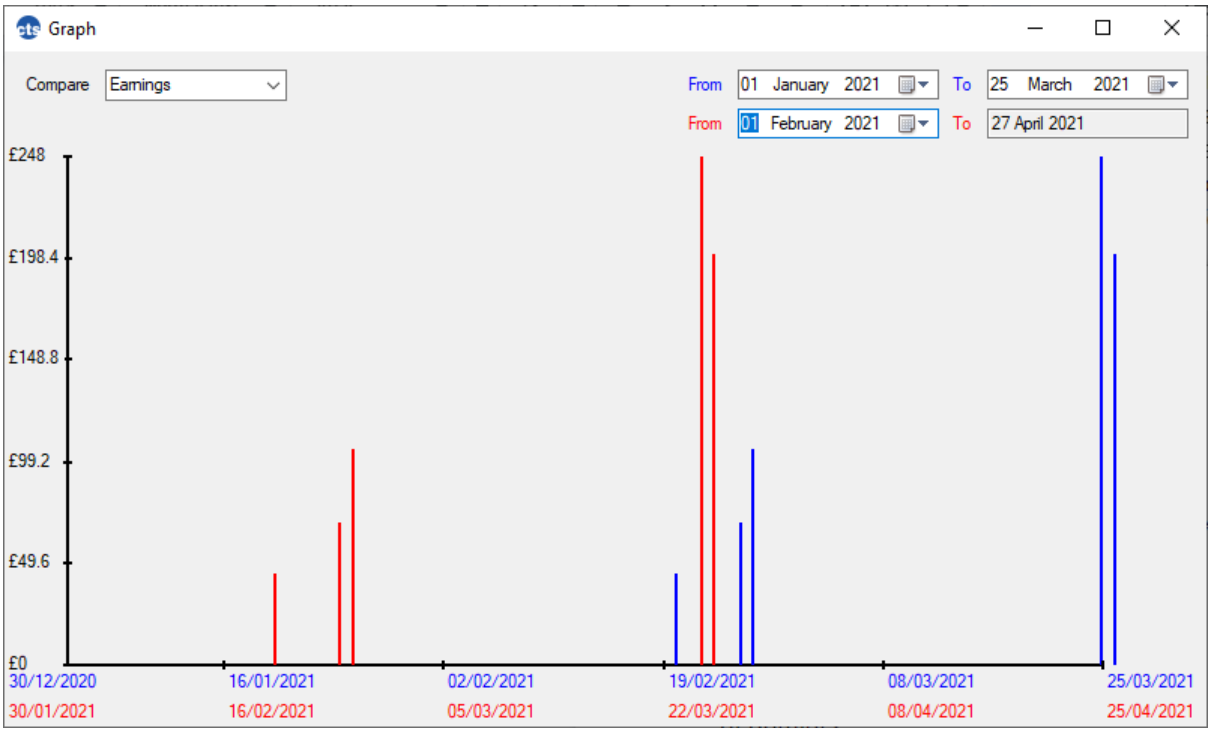


This user interface is very friendly, as the items are displayed in nice, organised, easy-to-read lists, that makes use of a search algorithm to find specific items without having to scroll through the many possible items to find it. This solution allows for records to be edited, should information need updating (for example, somebody changes address), and can be deleted should a customer die or no longer use the services of the company. The file can be cleared, should the director want to reset the system to a clean slate in the event that the files become corrupted for some reason. This system allows the user to access confidential information, such as customer and user details,

as well as having potentially devastating consequences if files are wiped without being backed up, and so access to these functions are restricted to authorised and trusted personnel only, who are granted the user access level of administrator (as previously discussed). This system functions correctly and is good for the user. However, a downside to this form is that data can be deleted and cleared and therefore lost forever. Some form of automatic back-up system could be implemented in a future version to minimise the risk of this. Not including this was an oversight during the design phase of the project, that upon reflection would have been a very good addition to the solution. Should I ever work on the project again for the client, or work on a similar project again in the future, I will be sure to learn from this oversight and include back-up and recovery systems as they are invaluable. However, although this system would benefit from such a feature, it is not necessarily handicapped by not having one,provided thesystemisused responsibly and the encrypted login system is enough to deter unauthorized access.

This system significantly increases the speed at which records can be found, as the system can find items via a search algorithm in near-instantaneous speeds, whereas manually searching for the data in the paper system requires employees to search through multiple paper books full of information that is not even alphabetised or sorted, and so every item must be looked at individually, which can take several minutes.

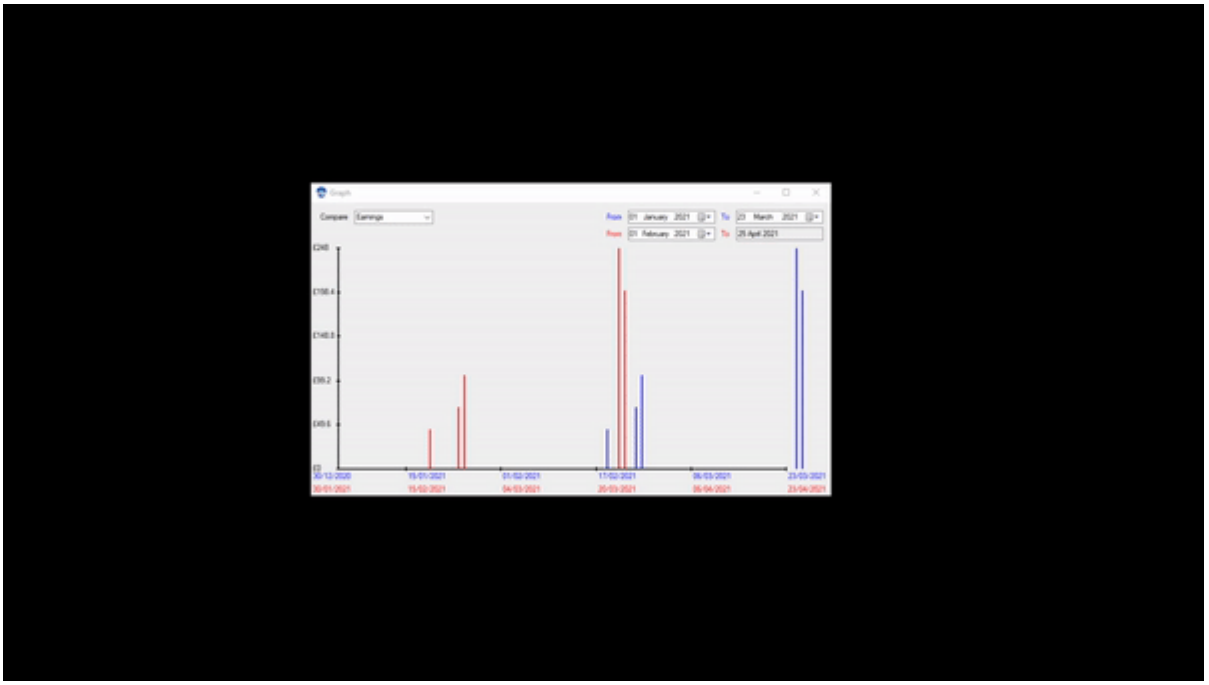
The final part of this application is the graph outputs, as shown below.

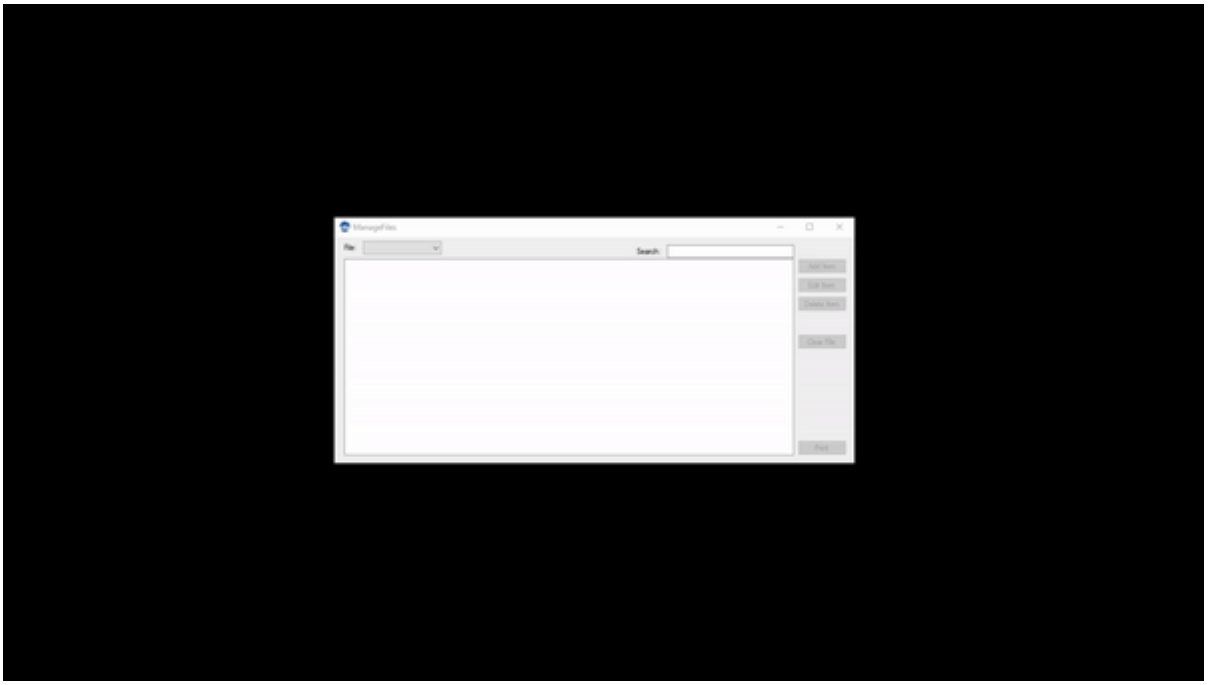


The purpose of these graphs is to allow the user to easily view the earnings over a user-defined period of time, using a barchart graph to display the values in an easily-understandable way. This allows the user to not only view but compare financial data, which is important as it will allow the user to see if there are any spikes / drops in earnings around certain times of the month / year, and so the director can make beneficial decisions using this information, such as working longer on the high-income days to maximise the increased revenue of these days. A downside of this graph is that there are no grid-lines or other guides and so the exact position of the bars can be slightly unclear, especially when working large numbers of data. This was an issue that was not realised until after production, when the graph had been completed. Another solution to this problem could be displaying information similarly to the timetable (see above) when a bar is selected, giving exact information when needed, but still providing good enough detail at a glance. In a future update I will learn from this mistake, and allow the user to toggle grid lines onto the graph should they want them, using a tickbox, and allow the user to see exact information by selecting a bar. Although the system is not perfectly precise due to this, the graph still serves its purpose and values can be interpolated from the graph. I would also add more data options to compare inthe future, however due to time constraints I was limited to only comparing the earnings. I did however create the graph calculations modularly, and so these additional comparisons can be added quite easily, without having to perform a large code rewrite, as it is already a modular system. For the graphs, I chose to create my own class as opposed to using the inbuilt libraries, as I had already done the mathematics behind the graphs during design, and so it was easy for me to use my own class where I had complete control over how the graphs worked, as all I had to do was implement my already existing design, instead of having to adapt it to work with the library's graph functions. This also means that in the future my graphs can be expanded upon to greater extent as I can write new functions for new features, whereas using a pre-existing library limits the graph to the capacbnilitiesofthat library, and so if I wanted to add a new feature that he libraries could not facilitate I would have the create the *whole* graph class from scratch, which is not ideal.

One feature on both of these forms is allowing the user to resize the form, and scaling and anchoring the controls to fit the new window size. This is useful on the graph, as it allows the bars to be spaced out and not squashed together, allowing the user to easily see the details, and for the manage files form, as it

will allow the user to fit more items on the screen at once, and so will be able to see more data without having to scroll, should they wish. Below are demonstrations of these features.



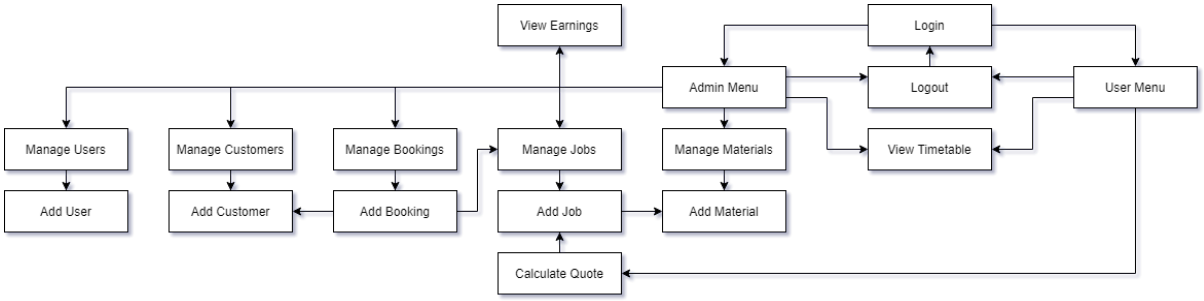


Another shortcoming of my project is that I was not able to successfully implement a printing solution into the application, which was one of my objectives, and requested by the client. This objective was not achieved, due to time constraints. I spoke to the client and explained that due to complications it may be necessary to not implement this feature at release in order to meet the deadline. I was assured that this was okay, as he can still

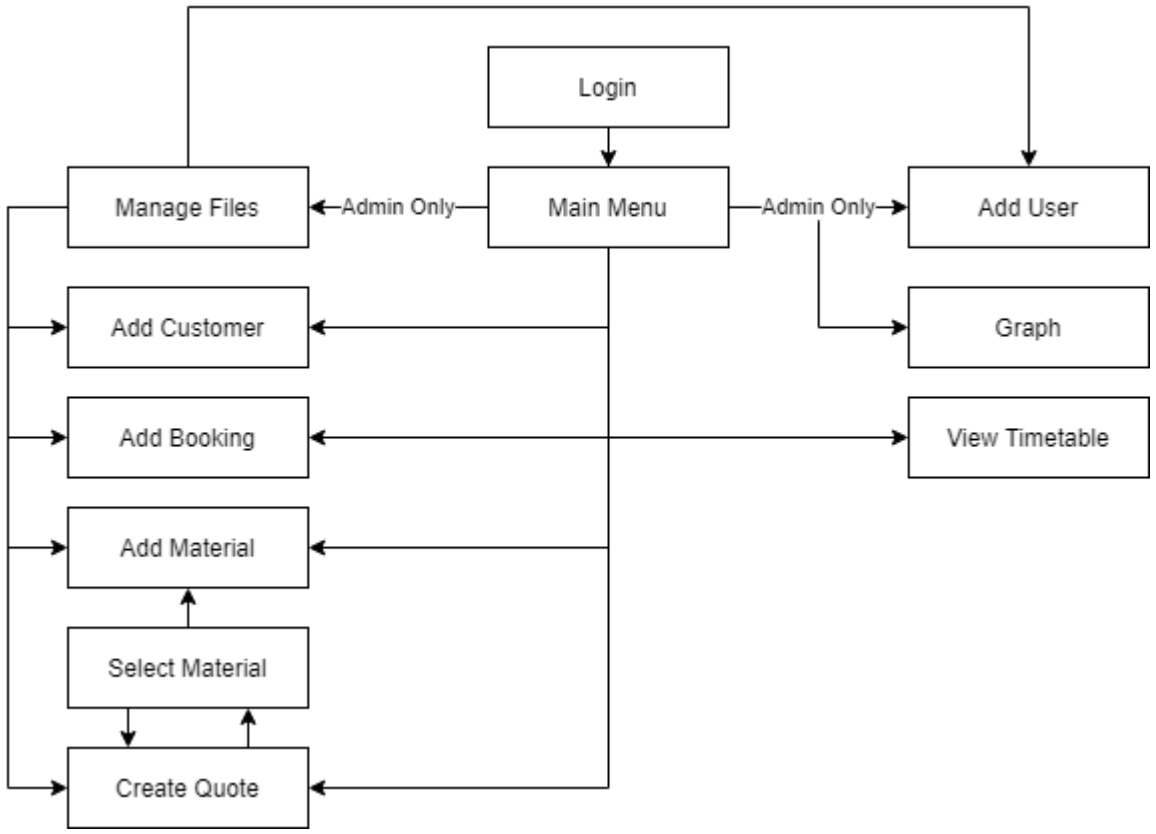
create hard copies of timetables by taking screenshots, and printing those, and so losing this feature was not a major loss. It is planned to add this feature in a future update.
Another minor downside to this solution is that the program is not the most memory efficient in some instances, for example, in some calculations arrays are created to temporarily hold data whereas variables could be used, but using whole arrays was an easier solution programming-wise, as well as being more time efficient, which was a key objective of this project. In future versions, many aspects of the code can be reviewed and refined into more efficient solutions, as some of the functions, whilst working and successfully performing their jobs, are a bit 'barbaric' and not as 'elegant' as they could be. An example of this is where sometimes iteration has been chosen over recursion solely for simplicity's sake. If there wasn't a strict time constraint on the development, I would spend more time refining the code before deployment. However, the importance was making the code work, and refining it (whilst useful) was not a priority and can be done in future iterations.

Objectives Analysis (Overview)
Below is a brief overview of the objectives established in the investigation, and how they were achieved in development, or, if they were not achieved, why.

Alterations to the Design's Tree Diagram



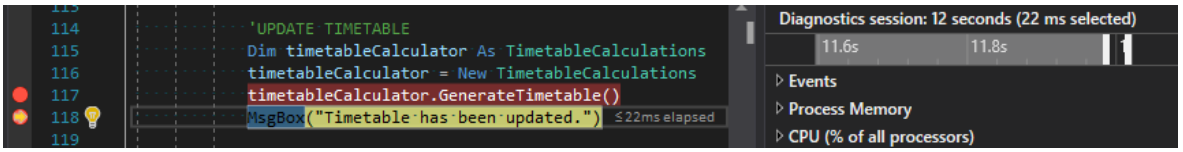
(Tree diagram from design phase)

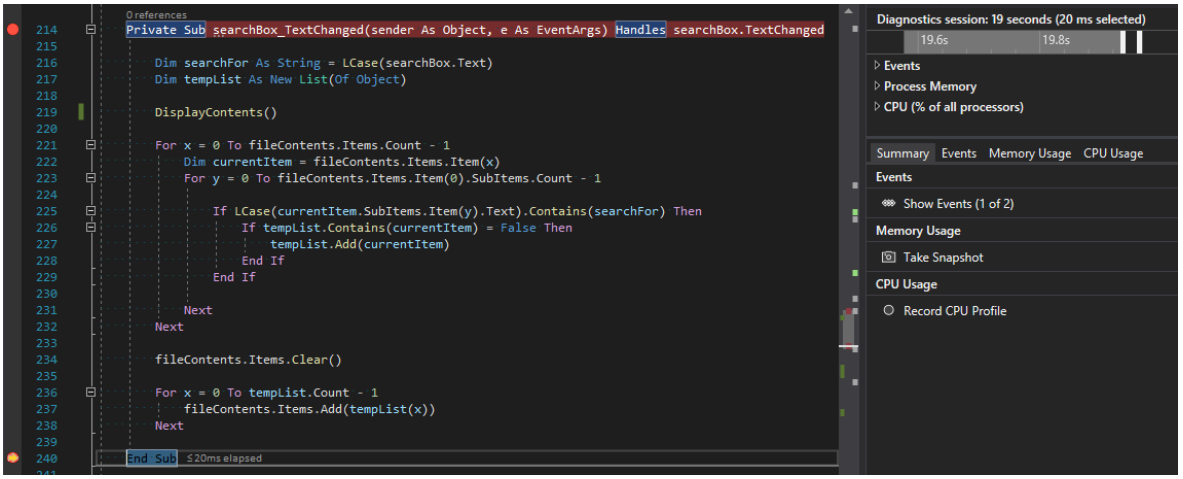


(Tree diagram of actual product)

As can be seen in the images above, the form navigation of this application has changed over the course of this project. Initially, the plan was to use 17 forms, however in the end this project was refined into 11 forms **without sacrificing features**. This was done as originally each file management form was going to be a unique form that made use of shared functionalities. However, in the end this approach was not taken and instead these forms were changed into a single form, that made use of a drop-down combobox that allows the user to select the file they wish to manage. The Add Job and Create Quote forms were merged into a single form, allowing the user to calculate the prices in the same form they could save the jobs in, as this reduced the amount of data input (as the input data for these forms are identical, and so would have to be input twice if they were separate). The admin and user menus were merged into form, making use of the control dispose function to delete controls for normal users preventing them from having access to administrative features by removing access to these forms. The final change was that the logout form was not implemented, as this was an unnecessary layer of complication, requiring users to sign out, and instead exiting the application is performed simply by closing the program using the default exit button in the top right corner (in the same way professional applications such as Google Chrome are exited).

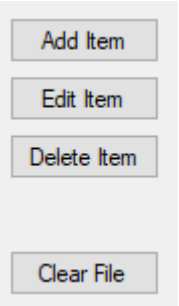
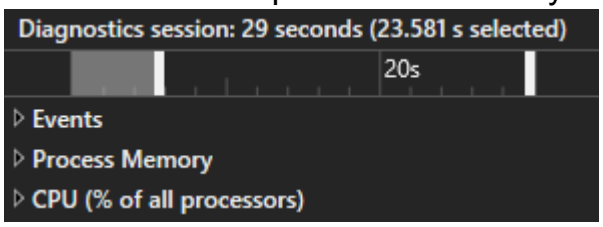
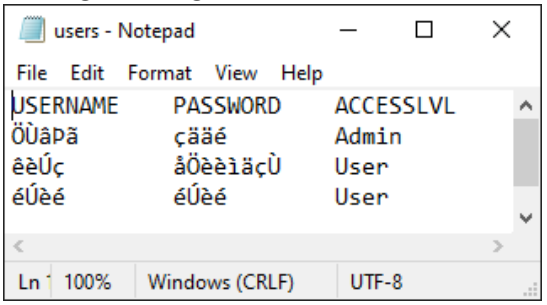
These changes were made to keep the simplicity of the application, as well as being faster by reducing ‘the number of hoops users have to jump through’ in order to perform their tasks, as simplicity and speed were core components of our vision for the application. By merging several forms, the program is also slightly smaller to store, as less forms and code in those forms are stored, which is another upside to this refinement.

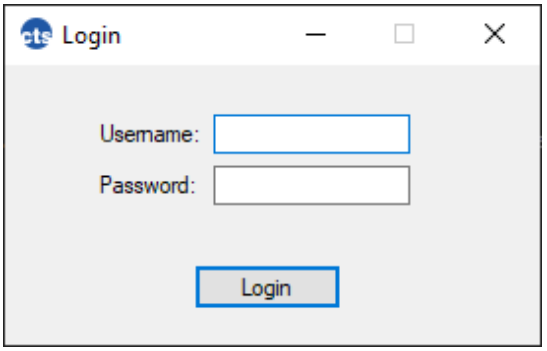
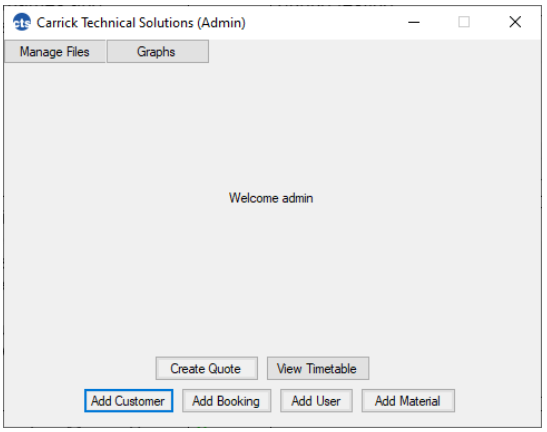
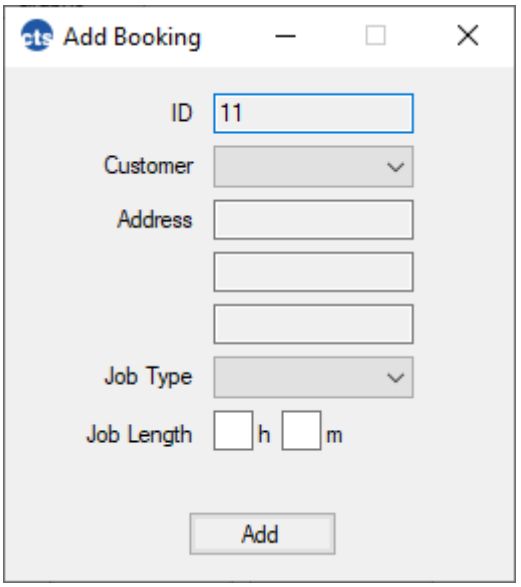
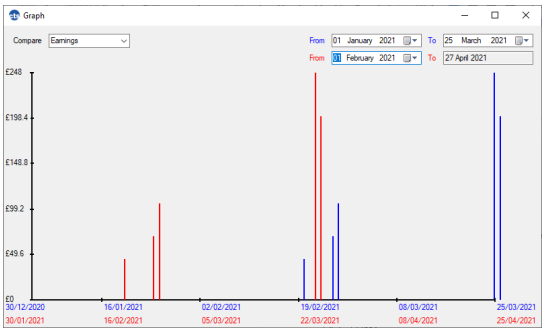
No.	Desc.	Done?	
1	It should be significantly faster to generate a timetable.	Yes	The timetable generation took around 22ms to complete when testing, which is significantly faster than Shaun’s half-an-hour manual process.
 Above is the code responsible for triggering the timetable generation. This screenshot shows that the time between these breakpoints (just before the process began, and after it was completed) took a time of 22ms on my computer. This time will vary depending on several factors, such as the specs of the computer, and the amount of bookings the algorithm has to handle, however, from my investigation I found that the old system could take Shaun over half an hour to perform, meaning even if his device cannot perform this task as fast as mine can, it is almost guaranteed that it will be faster than the old system. Another note is that as well as being faster, this solution is also personally easier for Shaun, as the computer handles all the calculations, and all he has to do is press a button.			
2	It should be faster to find a customer’s details.	Yes	Using the search bar feature in the Manage Files form took around 20ms during testing, whereas searching through the paper records could take several minutes.



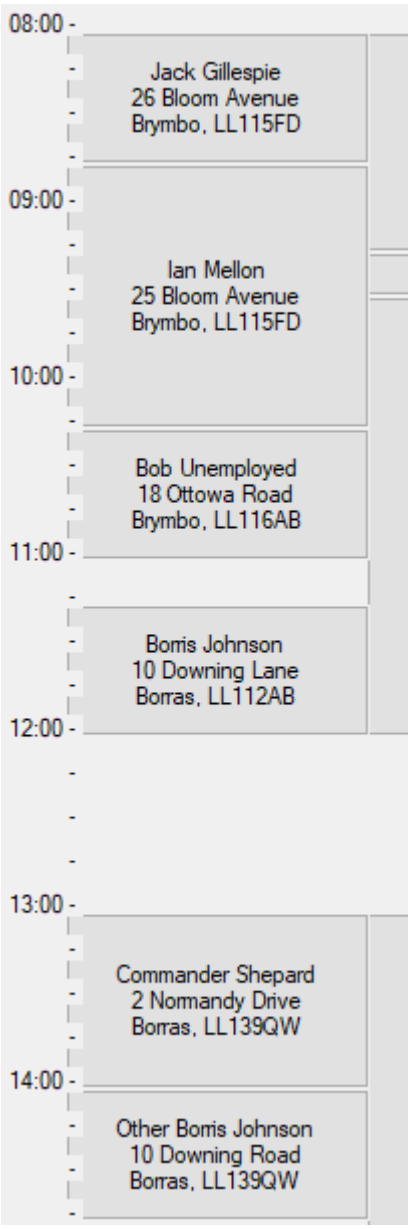
The above screenshot shows that it takes a total of 20ms to run the whole search function. According to my investigation research, it could take several minutes for the employees to search through the paper records and find the correct customer, which is a substantially longer amount of time.

3	Projected earnings for the week should be automatically generated.	Yes	 The timetable form automatically calculates the projected earnings for the selected week, and displays them using a text box, as shown below.
4	Information should be automatically checked for errors, such as words for numerical values.	Yes	Applicable validation routines are applied to all data-input fields where necessary. This is well documented and evidenced in the testing section of the code.
5	All staff should be able to access the required, and only the required, features and functions of the system.	Yes	This feature has been previously discussed and is also evidenced during the testing section of the project.
6	The director must be able to create, manage and delete user accounts.	Yes	Admin users (such as the director) are able to perform these functions using the following buttons on the Manage Files form:

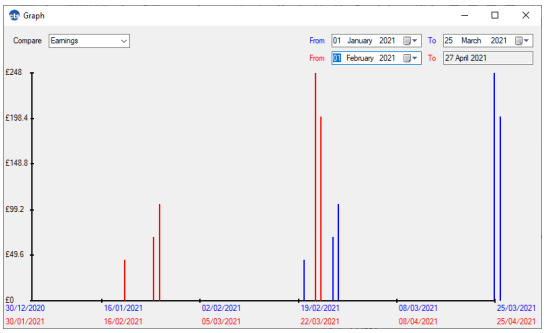
			
7	Users should be able to add entries and information easier.	Yes	This varies depending on several factors, however, on average the digital solution was fasterby an average of 17.8 seconds per item.
The time it takes to perform this task varies depending on the user, and both their level of digital literacy and physical writing speed. From some testing, it took on average 30.4 seconds for a customer to be added to the system (timing from the form opening, to the writing completing), whereas I found that it took an average of 48.2 seconds to record these details manually. Below is an example of one of the system tests: 			
8	All users must have securely encrypted usernames and passwords.	Yes	This has been well documented previously in the evaluation, and during testing. 

9	Login Screen. Username and password should be typed using a keyboard, and submitted using a mouse to interact with a button.	Yes	
10	Navigation Menu. The user should navigate to the required menu using the mouse to select the corresponding button.	Yes	
11	New Booking. The customer's details (name, address, postcode, etc.) should be entered using a keyboard. Some details, such as 'type of job' should be selected from a dropdown menu using the mouse.	Yes	
12	Financial Menu. The director should be able to select dates and filters through the use of the mouse.	Yes	

13	Presence checks should be used on almost all data inputs (barring optional inputs, and inputs where there is a default value, so it cannot be blank).	Yes	See testing section.
14	Data type checks should be used on all numerical values that are needed for calculations, such as quantities, prices, etc,	Yes	See testing section.
15	Format checks should be used on patterned values, for example postcodes should match the AA000AA format, and phone numbers should match the 07xxxxxxxx format.	Yes	See testing section.
16	Length checks should be used on all file-stored inputs, to ensure that the entries fit into the records.	Yes	See testing section.
17	Two-pass verifications should be used when creating new users, to ensure that the correct passwords are entered.	Yes	See testing section.
18	Exclusive checks should be carried out when new users are created to ensure that the username in question is not already taken.	Yes	See testing section.

19	Timetable should be generated to ensure that customers are ordered based on their geographical proximity.	Yes	 This image shows that the algorithm has correctly sorted the bookings by their location, as the first group of items are all Brymbo addresses, with the second group being Borras. As well as this, the items are sorted within their area groups by postcode, which can be seen by the fact that bookings with the same postcodes are next to each other.
20	Quote prices should be calculated from the input data.	Yes	 The quote calculation job successfully calculates the hourly

			rate and resource costs of quotes.
21	New user data should be encrypted.	Yes	See objective 8.
22	Projected earnings from the week's jobs should be calculated.	Yes	See objective 3.
23	Past earnings should be comparable and searchable.	Yes	 The timetable form automatically calculates the confirmed earnings for the selected past week, and displays them using a text box, as shown above.
24	Timetable should be viewable by all staff, as a visual representation.	Yes	Regardless of whether the user is an admin or standard user, they are able to access the timetable via the main form.
25	Timetable should be printable.	No	This objective was not achieved, due to time constraints. I spoke to the client and explained that due to complications it may be necessary to not implement this feature at release in order to meet the deadline. I was assured that this was okay, as he can still create hard copies of timetables by taking screenshots, and printing those, and so losing this feature was not a major loss. It is planned to add this feature in a future update.
26	Quotes should be viewable by the director. Results of calculations should be viewable by the director.	Yes	This can be achieved using the manage files form and selecting 'Jobs.txt'.
27	A list of users should be viewable by the	Yes	This can be achieved using the manage files form and selecting

	director.		‘Users.txt’.
28	A graph should display the earnings of days between two sets of dates, where they can be directly compared.	Yes	

My Own Performance

Throughout this project, I have encountered many challenges and problems which I have had to overcome through perseverance and creative problem solving. Through this project I have improved my ability to program and create forms using Visual Basic and the Visual Studio Windows application visual IDE. Not only has this improved my ability to program using the Visual Basic language, but I have also gained skills that are applicable to all programming languages that can help me in future projects regardless of which language I am using. This project has also taught me to better improve my time management, as I has to learn when to stop working on a feature if I hit a major roadblock and work on another aspect of the project, and come back to a solution, as opposed to going down the rabbit hole of struggling with a single aspect until it is solved, resulting in crunching being required to catch up in other areas. This project also improved my problem solving and critical thinking skills, as I had to create solutions for many issues (some of which are documented in the developmental testing section).

One of my primary strengths in this project was the development side, as I come from a strong programming background, and so am both experienced using programming languages and tools, but also thoroughly enjoy the process and so was never lacking for motivation to work on my code. Because of this experience I was able to utilise complex features such as multiple recursive functions for the timetable generation function of the code (a flowchart of which can be seen in the design section). Due to my logical experience, I was able to complete the design without encountering many issues, as I was able to work out the flowcharts and pseudocode functions due to my knowledge of programming and what is possible within the languages. I was also able to understand some of the more abstract and

complex concepts such as using fixed length fields, and calculating the positions of characters within the files, especially since I have done a very similar process before using C# to create save files for games. The only hurdle I encountered for this was not knowing the Visual Basic syntax for the functions, however using the Microsoft documentation and programming forums such as StackOverflow, I was easily able to find the VB versions of my C# keywords and implement them into my code. The greatest programming issue I encountered was issues with the timetable generation process, as one error would follow another, meaning as soon as one bug was resolved, another one would need fixing. Some of more complex errors encountered in this class were difficult to debug due to the complexity of the task (consisting of many intertwined functions), as well as it making use of recursive functions, and some of these issues and my solutions to them can be seen in the developmental testing section of the code.

I did, however, struggle with the large writing sections of the project, such as the investigation and evaluation, as unlike with coding I can struggle to find motivation for long writing tasks, and can often procrastinate them,which is not beneficial. I had to teach myself to work through discipline and not purely through motivation in order to allow me to complete these tasks on time, whilst still in depth and detailed. I also struggled with the developmental testing slightly, as I can get so caught up in following error codes and determining the genesis of bugs, and working to solve them, I would often forget to slow down and document the process instead of simply powering through and solving them. The investigation was particularly difficult at times, as the interview and observations required me to communicate with others, which is not my strong suit, especially since I can struggle to converse about complex programming terminologies due to an inexperience of explaining and discussing such things with non-programmers. This pushed me out of my comfort zone however, and was an invaluable experience for me, as it allowed for me to hone these weaker skills of mine.

I approached this project by abstracting the problem into smaller problems that I began to solve individually and systematically. This allowed me to not be overwhelmed by too many issues at once, by taking the whole project ‘one small step at a time’. An example of this was the timetable calculations. This whole process is quite complex and complicated, and so during my design I broke down the process into simple steps, such as grouping, sorting and then slotting. This allowed me to approach the one problem at a time and find a solution for it, as opposed to attempting the whole process at once. This was

beneficial as it allowed me to truly pinpoint the core components of the problem, and so I could create very specific and accurate solutions.

Another approach I used was by developing multiple unrelated aspects of the project simultaneously. For example, as I was working on the graph's generation, I was also working on the user access levels. This allowed me to stop working on one feature when I became stuck (for example, on the graph), giving me a break from that problem, and yet during this 'break' I would work on the other aspect (the access levels), and so this prevented me from becoming overworked and stressed by sticking on the same problem for too long, and yet I was also still being productive during these 'breaks' as I was still working on the application, just another part of it.This solution was highly beneficial for productivity, and reduced the amount of times I would become stressed or overwhelmed by a problem, whereas in past projects where I haven't implemented this technique, I have struggled with becoming overwhelmed.

As I was working on the project, I would be sure to read over the investigation and design at regular intervals, to ensure that the client's needs,and my own vision, were fresh in my mind. This prevented me from straying away from the actual goals inpursuit of non-essential details.

Overall, this has been an invaluable learning experience, as not only have I improved my skills and knowledge, but also I have identified methods and techniques that did and did not work well, and so can employ these good strategies and avoid these downfalls in future projects.