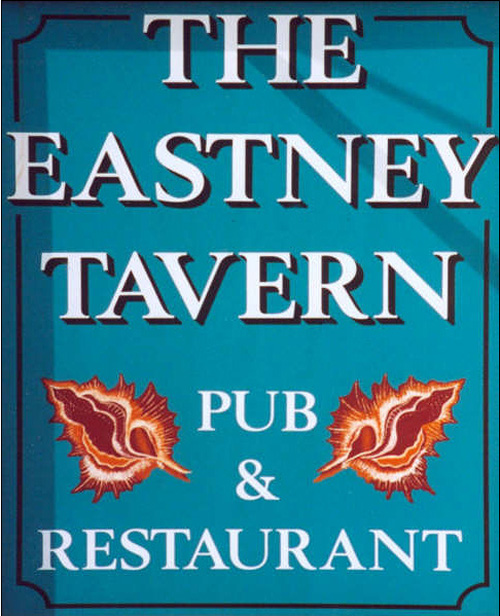
Pub

Ordering and analysis

Computer System



Joseph Laithwaite

Table of Contents

[Analysis 6](#_Toc58333363)

[Background to the project 6](#_Toc58333364)

[Identification of the problem 6](#_Toc58333365)

[Description of current system 6](#_Toc58333366)

[Identification of the prospective user 7](#_Toc58333367)

[Identification of user needs 7](#_Toc58333368)

[Acceptable limitations 7](#_Toc58333369)

[Areas Not Included 7](#_Toc58333370)

[Future Development 7](#_Toc58333371)

[Software 7](#_Toc58333372)

[Hardware 8](#_Toc58333373)

[Time 8](#_Toc58333374)

[User Knowledge 8](#_Toc58333375)

[Data source 8](#_Toc58333376)

[Data destinations 10](#_Toc58333377)

[Data Volumes 11](#_Toc58333378)

[Analysis Data Dictionary 13](#_Toc58333379)

[Data flow diagrams (DFDs) 17](#_Toc58333380)

[Existing system 17](#_Toc58333381)

[Proposed System 18](#_Toc58333382)

[Entity relationship (E-R\_ model (if appropriate), E-R diagrams, entity descriptions 19](#_Toc58333383)

[System Objectives 20](#_Toc58333384)

[Qualitative Objectives 20](#_Toc58333385)

[Quantitative Objective 20](#_Toc58333386)

[Realistic appraisal of the feasibility of potential solutions 21](#_Toc58333387)

[Justification of chosen Solution 22](#_Toc58333388)

[Evidence of use of appropriate analysis techniques 22](#_Toc58333389)

[Interviews 22](#_Toc58333390)

[Questionnaire 24](#_Toc58333391)

[Source Documents Analysis 25](#_Toc58333392)

[Design 30](#_Toc58333393)

[Overall system design 30](#_Toc58333394)

[Modular structure of system 31](#_Toc58333395)

[Design Data Dictionary 31](#_Toc58333396)

[Algorithms for data transformations 37](#_Toc58333397)

[Login 37](#_Toc58333398)

[Saving Data (employee and meal forms) 38](#_Toc58333399)

[Meal selection 38](#_Toc58333400)

[Search Employee file 39](#_Toc58333401)

[Delete employee record 39](#_Toc58333402)

[Reports 40](#_Toc58333403)

[Orders 41](#_Toc58333404)

[User interface design (HCI) rationale 42](#_Toc58333405)

[UI sample of planned data capture and entry design 44](#_Toc58333406)

[UI sample of planned valid output designs 50](#_Toc58333407)

[Description of measures planned for security and integrity of data 52](#_Toc58333408)

[Description of measures planned for system security 52](#_Toc58333409)

[Overall test strategy 53](#_Toc58333410)

[White Box testing 53](#_Toc58333411)

[Black Box testing 53](#_Toc58333412)

[Unit testing 53](#_Toc58333413)

[Integration testing 53](#_Toc58333414)

[Preliminary test plan & Test data 54](#_Toc58333415)

[Meal input screen 54](#_Toc58333416)

[Employee Input screen 56](#_Toc58333417)

[Orders 58](#_Toc58333418)

[Reports 60](#_Toc58333419)

[Technical Solution 62](#_Toc58333420)

[Login Form 62](#_Toc58333421)

[Menu Form 66](#_Toc58333422)

[Meal Form 69](#_Toc58333423)

[Orders Form 78](#_Toc58333424)

[Reports Form 94](#_Toc58333425)

[Employees Form 115](#_Toc58333426)

[Setting/Help Form 128](#_Toc58333427)

[Eastney Tavern Module 135](#_Toc58333428)

[Conversion Plan 138](#_Toc58333429)

[System testing 140](#_Toc58333430)

[Test Data 140](#_Toc58333431)

[Employee File 140](#_Toc58333432)

[Menu File 141](#_Toc58333433)

[Meal File 141](#_Toc58333434)

[Order File 142](#_Toc58333435)

[OrderItem File 142](#_Toc58333436)

[Alpha testing 143](#_Toc58333437)

[Employee Form 143](#_Toc58333438)

[Settings/Help 152](#_Toc58333439)

[Meal Form 161](#_Toc58333440)

[Orders Form 166](#_Toc58333441)

[Reports Form 175](#_Toc58333442)

[Navigation Test 184](#_Toc58333443)

[Print Testing 185](#_Toc58333444)

[Beta testing 186](#_Toc58333445)

[Acceptance testing 190](#_Toc58333446)

[System maintenance 192](#_Toc58333447)

[System overview 192](#_Toc58333448)

[System Development 192](#_Toc58333449)

[System Structure 193](#_Toc58333450)

[A sample of detailed algorithm design 195](#_Toc58333451)

[Employee Search 195](#_Toc58333452)

[Delete Order 198](#_Toc58333453)

[Procedure and variable descriptions 201](#_Toc58333454)

[Eastney Tavern Module 201](#_Toc58333455)

[Login Form 205](#_Toc58333456)

[Menu Form 205](#_Toc58333457)

[Meal For 206](#_Toc58333458)

[Employee Form 207](#_Toc58333459)

[Orders Form 208](#_Toc58333460)

[Reports Form 210](#_Toc58333461)

[Settings/Help Form 214](#_Toc58333462)

[Annotated listings of program code/macro code and tailoring 214](#_Toc58333463)

[Appraisal 215](#_Toc58333464)

[Comparison of project performance against numbered general and specific objectives 215](#_Toc58333465)

[Qualitative Objectives 215](#_Toc58333466)

[Quantitative Objectives 216](#_Toc58333467)

[User feedback authenticated by assessor 218](#_Toc58333468)

[Analysis of user feedback 220](#_Toc58333469)

[Possible extensions 221](#_Toc58333470)

[User Manual see other document 222](#_Toc58333471)

# Analysis

## Background to the project

The company I will be creating this project for is the Eastney Tavern, which is a pub in Eastney close to the sea in Portsmouth. It is an old pub, which last changed ownership in 2011. It serves a large selection of drinks and also contains a restaurant which serves a variety of classic pub food such as fish and chips, roast dinners and Burgers as well as a variety of curries, salads and much more. The owner of the pub is Chris moon (the client) and the head chef is Darren Mundey.

They employ around 30 people (this number often changes with people leaving to/from university and other jobs), the jobs span bar work, waiting, cooking and cleaning. Many of these employees work zero hour contracts and get paid cash in hand.

## Identification of the problem

There are a number of problems with the current system from the wrong food being ordered due to poor handwriting and miscommunication, a non-existent employee and supplier database, Customers over or under paying for food due to waiting staff’s poor math’s and a non-existent way of analysing different meals and dates.

## Description of current system

Currently the system contains many different pieces of paper and a range of disjointed systems, this in turn causes a variety of mistakes. During service waitresses/waiters write down the orders on a piece of paper and bring that into the kitchen to give to the chefs, this works fine but due to human mistake (handwriting) sometimes the wrong meal or wrong number of meals are cooked. Another problem is if one meal runs out and the chefs tell on of the waitresses this information is sometimes forgotten and allows people to order meals, which can’t be cooked causing confusion and disruption. At the end of service a chef has to count up all the main meals sold which is then written on a large year planner and shows a rough guide to how well the pub does on different days.

Another part of the pubs various system is the storage of personal information such as phone numbers and addresses for employees and suppliers. The suppliers are stored in the owner’s contacts on his phone and on various pieces of paper in a file in the kitchen. The staff information is then split up with the head chef having all the kitchen staffs information in his phone and the head waitress having all the front of house staff stored in her phone. This can cause large problems when people are sick or on holiday.

Finally to pay the waiting staff have to add up the cost of the meal in there head (or with a calculator), this can take longer than necessary and cause easily avoided mistakes causing customers to over or under pay.

## Identification of the prospective user

The users will all be part of the restaurant in the Eastney tavern including the waitresses/ waiters and the chefs. This product will be useful for many aspects of the restaurant. The main users will be the waiting staff as during service they will use the system to order food, Print receipts and check additional information. The only other main users will be the head chef who will edit the database of meals when new dishes are added to the menu or costs change. The boss will also be a user and will mainly use the system to analyse what meals are selling the best. The head chef and head waitress will also use the system to store employee data and recall it when they need to contact employees.

## Identification of user needs

There will be a large variety of users of this system, some have relatively good I.T skills but others do not, the number of waiting staff and how they will need to user the system very quickly during service means the ordering system I produce will have to be very simple to use and quick to understand.

This said I can make the analysis sections more complicated as only the head chef and boss (who both have good computing skills) will need to use this.

## Acceptable limitations

### Areas Not Included

I haven’t included an Ingredients file and goods supplied file, this was something discussed which would allow a meal to have all the ingredients (and suppliers ID for the ingredients) included in the system. We thought this would be a good way for easy shopping and ordering lists to be created and split up by supplier. On top of this a suggestion of automatic emailing to the suppliers from within the system would be a great addition, although at this time completing these ideas would take too much time and the e-mailing would be outside of my current ability as a programmer.

### Future Development

If I had more time I would definitely include an ingredients and a goods supplied file, this would mean in the meal tab you can tick a radio box called restock and then in a complex report all the meals which need restocking would have their ingredients and quantities listed in a report.

### Software

The current computer, which I will be using, is running windows XP. So my solution will have to run well on XP, as well as a new version of windows if the client decides to upgrade the system (especially now windows no longer supports XP). This shouldn’t cause a problem as all visual basic

### Hardware

The current hardware is a DELLPRECICION 380 computer with a 64 bit Pentium dual core processor, 2 GB RAM and 250GB storage, they also have an A4 printer. I plan to buy two Receipt printers with the £150.00 budget the client has set out. I think I will purchase two Epson TM T88II and these will cost around £80 for both.

### Time

The overall deadline for the solution to be complete is the 10th of February 2015. However I have imposed many sub deadlines to ensure the completion of the project and provide information to the client.

### User Knowledge

There are a variety of different users of this system which will constantly be changing due to new staff being hired and others leaving, these all have a varied amount of computer knowledge. This means when creating my system I will have to make it very basic and intuitive to allow users to pick up and use the system with little to no training. I will also be creating a user manual which will be available in a help file in the system. This will outline how to you use each part of the system so if people struggle to understand part of the system they will be able to read this file should clear up the problem.

## Data source

The head chef or head waitress when adding new or editing information will enter most of the data through a keyboard and will be the soul inputters for most of the data. The majority of this data will be entered only once when transferring from the old system to the new one. These are the data fields:

Meal

Meal ID – The number will be automatically generated when adding a new meal

MealName – The name of the meal used to identify it by users

Menu – This will help find meals more quickly as you can search inside a given menu (dessert, lunch, Evening, Sunday etc.)

Cost – This is the price it costs to produce each portion of the meal

Price – The price the meal is sold to customers

Supplier ID – This is the ID of the supplier who supplies the meal

Allergy – This is advice for waitresses about if the meal has any allergy advice (contains nut, milk etc.)

InStock – Radio button

HistoricMeal – Radio button

Employee

Employee ID – This number will be automatically generated when adding a new employee.

EmployeeName – The name of the employee, this will be printed on receipts

PhoneNumber – Used to contact the employee

Email - Used to contact the employee

Address - Used to contact the employee

Job – The job description (waiter, chef, bar staff etc.)

PayRate – The rate of pay per hour, will be used to help calculate pay (not by the system)

EmergencyContactName

EmergencyContactNumber

All waiting staff users will enter this data

Order

Order ID - Automatically generated number

TableNumber – Entered using a keyboard by user

Date – Entered by the system using the current date

EmployeeFirstName – entered by the system and worked out using the employee ID

OpenOrder – Radio button allowing users to see which orders have been paid for and which haven’t

OrderItem

Order ID – Referenced to The order ID in Orders

Meal ID – found by the system after a meal name is entered

Quantity – entered by user

Cost – making a copy of the current cost (from meal) to allow this system to work over time

Price – for the same reason as cost making a copy of the current cost (from meal)

## Data destinations

There will be a variety of outputs and these will all appear both on screen and also have the option to print the form, for use in physical filing and receipts.

The first two outputs are very similar and are produced during service. One is printed in the kitchen and the other at the bar and given to customers as a receipt.

Orders Printed report:

EmployeeFirstName 🡪 Order.dat

Date 🡪 Order.dat

Time 🡪 When printed

Table Number 🡪 Order.dat

MealName 🡪 OrderItem.dat

Quantity🡪OrderItem.dat

Receipt Printed Report:

EmployeeFirstName 🡪 Order.dat

Date 🡪 Order.dat

MealName 🡪 OrderItem.dat

Quantity 🡪OrderItem.dat

MealPrice 🡪 Meal.dat

Total excluding VAT 🡪 Calculated

Total Including VAT 🡪 Calculated

Only the Head Chef and boss will use the rest of the reports.

View all Meals

Per day (or group of days) ordered by Total Sales

Meal ID 🡪 Meal.dat

Meal Name 🡪 Meal.dat

Cost 🡪 Meal.dat

Price 🡪 Meal.dat

Total Sold 🡪 Accumulated from orderitem.dat

Total income 🡪 Worked out by total sold x (cost-price)

## Data Volumes

I used chapter 3 of the “Computing projects in Visual Basic.net “book to get the necessary information on volumetrics.

Meal File

|  |  |  |
| --- | --- | --- |
| Field | Data-Type | Storage Required (Bytes) |
| Meal-ID | Short (Numeric) | 2 |
| MealName | String (20) | 20 x 2 = 40 |
| Menu | String (15) | 15 x 2 = 30 |
| CostCurrent | Single (Numeric) | 4 |
| Price Current | Single (Numeric) | 4 |
|  |  |  |
| AllergyAdvice | String (30) | 30 x 2 = 60 |
| InStock | Boolean | 2 |
| Historic | Boolean | 2 |
|  | | |
| Total | --- | 144 |
|  | | |
| For 400 Records 🡪 400 x 144 = 57,600 🡪 57.6Kb | | |

Employee File

|  |  |  |
| --- | --- | --- |
| Field | Data-Type | Storage Required (Bytes) |
| Employee -ID | Short (Numeric) | 2 |
| EmployeeFirstName | String (15) | 15 x 2 = 30 |
| EmployeeSurnameName | String(15) | 15 x 2 = 30 |
| PhoneNumber | String(11) | 11 x 2 = 22 |
| Email | String(20) | 20 x 2 = 40 |
| Address | String(30) | 30 x 2 = 60 |
| Job | String(10) | 10 x 2 =20 |
| PayRate | Single | 4 |
| EmergencyContactName | String (30) | 30 x 2 = 60 |
| EmergencyContactNumber | String(11) | 11 x 2 =22 |
| Password | String(15) | 15 x 2 = 30 |
|  | | |
| Total | --- | 320 |
|  | | |
| For 50 Records 🡪 50 x 320 = 16,000🡪 16kb | | |

Order File

Average 20 tables per day and data must be kept for at least 7 years for tax.

20\*365\*7 = 51,100 records

|  |  |  |
| --- | --- | --- |
| Field | Data-Type | Storage Required (Bytes) |
| Order-ID | Integer | 4 |
| Table Number | Short (Numeric) | 2 |
| Date | Date | 8 |
| Employee ID | Short (Numeric) | 2 |
|  | | |
| Total | --- | 16 |
|  | | |
| For 51,100 Records 🡪 51,100 x 16 = 817,600b🡪 817.6Kb | | |

OrderItem File  
Average of 6 meals per table

51,100\*6 = 306,600 Records

|  |  |  |
| --- | --- | --- |
| Field | Data-Type | Storage Required (Bytes) |
| Order-ID | Integer | 4 |
| Meal-ID | Short (Numeric) | 2 |
| Quantity | Short (Numeric) | 2 |
| Price | Single (Numeric) | 4 |
| Cost | Single (Numeric) | 4 |
|  | | |
| Total | --- | 16 |
|  | | |
| For 306,600 Records 🡪 306,600 x 16 = 🡪 4,905,600b 🡪 4.9Mb | | |

Menu File

|  |  |  |
| --- | --- | --- |
| Field | Data-Type | Storage required |
| Menu Name | String(20) | 2 x 20 = 40 |
|  | | |
| Total | --- | 40 |
|  | | |
| For 10 records 🡪 10x 40 = 400b | | |

In total all the data inputs will need

57,600 + 16,000 + 817,600 + 4,905,600 + 400 = 5797200 bytes

5.8 Mega Bytes

## Analysis Data Dictionary

Meal File

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data |
| Meal-ID | Primary | Unique meal identifier | Whole Number | 1 – 300 | 135 |
| MealName | ––– | The name of the meal | Text | ––– | Fish and Chips |
| Menu | Foreign | A way to split the all meals into sub categories, such as starters, desserts, Sunday lunch, breakfast etc. | Text | ––– | Mains |
| CostCurrent | ––– | The cost of buying in a meal | Decimal Number | £1.00 – £30.00 | £3.50 |
| PriceCurrent | ––– | The price the meal is sold for | Decimal Number | £5.00 – £40.00 | £8.00 |
| AllergyAdvice | ––– | Information on allergy advice | Text | ––– | Gluten |
| InStock | ––– | A Boolean variable used so the waitresses can’t order an item which isn’t in stock | Boolean | Yes–No | Yes |
| Historic | ––– | A Boolean variable to remove a meal from the service choices but keep information so the system can be used over time. | Boolean | Yes–No | No |

Employee File

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data |
| Employee -ID | Primary | Unique ID for each employee | Whole Number | 0-50 | 32 |
| PayRate | ––– | The Hourly rate of pay for the employee | Decimal Number | £3.79 – £10.0 | £6.50 |
| Emergency Contact Number | ––– | The employees emergency contacts phone number | Whole Number | ––– | 077 8257 7828 |
| Phone Number | ––– | The employees contact phone number | Whole Number | ––– | 077 9282 6246 |
| Address | ––– | The employees home address | Text | ––– | 273 Jesse Road, Portsmouth |
| Employee SurnameName | ––– | Surname of the employee | Text | ––– | Biggins |
| Emergency ContactName | ––– | The employees emergency contacts name | Text | ––– | Janet Biggins |
| Employee FirstName | ––– | The first name of the employee | Text | ––– | Lucy |
| Email | ––– | The employees email address | Text | ––– | Lucy.Biggins@live.co.uk |
| Job | ––– | The Job title of the employee | Text | ––– | Waitress |
| Password | ––– | This will act as a security measure to stop unauthorized people accessing the personal and financial information. | Text |  | QwErTy321&\* |

Order File

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data |
| Order-ID | Primary | A unique identifier for each order | Whole Number | 0 – 51,100 | 563 |
| Table Number | ––– | The number of each table, first number represents section, second two numbers represent the exact table | Whole Number | 001-410 | 201 |
| Date | ––– | The date on the day of the order | Date | Any Date | 13/09/2014 |
| Employee ID | Foreign Key | The Employee ID of the waitress | Whole Number | 0-50 | 25 |

OrderItem File

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data |
| Order-ID | Primary | A unique identifier for each order. | Whole Number | 0-51,100 | 563 |
| Meal-ID | Primary | Unique meal identifier. | Whole Number | 0-300 | 135 |
| Quantity | ––– | The number of one item, which is ordered. | Whole Number | 1-20 | 3 |
| Price | ––– | The current price the meal is sold for. | Decimal Number | £1.00-£30.0 | £8.00 |
| Cost | ––– | The current price to by the order in. | Decimal Number | £5.00-£40.00 | £3.50 |

**Menu File**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data |
| Menu Name | --- | The name of a menu | Text | --- | Dessert |

## Data flow diagrams (DFDs)

### Existing system

Ordering

Ordering

Customer

Order

Receipt

Meal Name

quantity

Meal Name

quantity

Meal Name

quantity

Meal Price

Menu

Meal Name

Meal Price

### Proposed System

Meal ID

Quantity

Price

Cost

Meal Name

Meal ID

Employee ID, EmployeeFirstName, EmployeeSurnameName,

EmergencyContactName, EmergencyContactNumber

PhoneNumber,

Email, Address,

Job, PayRate,

Order

Table Number

Employee Name

Meal Name

Quantity, Price

Total Price, Date

Employee

Name

Order ID

Meal ID

Quantity

PriceCurrent

CostCurrent

MealName, Menu

CostCurrent, PriceCurrent,

Allergy, InStock, HistoricMeal

Meal ID

MealName

Menu

CostCurrent

PriceCurrent

Allergy

InStock

HistoricMeal

Table Number

Employee Name

MealName

Quantity

Date

Order ID

Table number

Date

Employee ID

Meal

User (waiter)

Receipt

Order

Meal Name

Table Number

Meal Name

Quantity

Employee ID

Meal

Name

Meal ID

CostCurrent

PriceCurrent

Order

OrderItem

User (Chef)

Enter meal data

Employee

Employee ID

User (Admin)

EmployeeFirstName, EmployeeSurnameName, PhoneNumber, Email, Address, Job, PayRate, EmergencyContactName, EmergencyContactNumber

Enter Employee Data

Order

Order –ID

Employee ID

Date

## Entity relationship (E-R\_ model (if appropriate), E-R diagrams, entity descriptions

Employee Info

Employee

Meals by Most Popular

Meals Most popular

Employee Performance

Employee Performance

Orders

Order Items

Employee

Meal

Takes

Customer

Gives

Menu

Meal (Meal ID, MealName, Menu, Cost, Price, Allergy, InStock, Historic)

Employee (Employee ID, EmployeeFirstName, EmployeeSurnameName, PhoneNumber, Email, Address, Job, PayRate, EmergencyContactName, EmergencyContactNumber)

OrderItems (Order ID, Meal ID, Quantity)

Orders (Order ID, TableNumber, Date, EmployeeName)

Menu (MenuName)

## System Objectives

Through my investigation I have discovered many aspects of the current system that should be included in the new one, some improvements to these and some totally new functionality that the client wants. Here I’ve specified exactly what the client wants me to build and what I’m promising to make. This list is split into two. Qualitative objectives are more open ended and subjective, these will have to be agreed with the client whether or not I’ve met them. Quantitative objectives are more definite and can be tested by myself, as they are a yes or no statement.

### Qualitative Objectives

1. The system should load quickly and run bug free.
2. The program should be very easy to use; it should have an intuitive user interface which is consistent throughout the system. The system should be simple enough that everyone no matter their computing skills can use the system.
3. The system should have a login system, allows most employees only access to the orders form and only allow administrators access to changing data and analytical tools.
4. The system should automatically calculate the price for each order.
5. The system should calculate the change needed when paying.
6. The program should produce many useful printed reports.
7. The system should contain a lot of dropdown boxes to minimize the amount of free text entered, this is to reduce errors and speed up data entry
8. This project needs to be completed by Tuesday the 10th of February 2015.

### Quantitative Objective

1. The system must allow users to enter many meals on one order and the system will be able to automatically add up the total.
2. The system must store records for at least six years for tax reasons.
3. The system should print out the order information on two printers one as a receipt and the other as an order for the kitchen.
4. The system should be easy to add important data about the meals such as; Meal-ID, MealName, Menu, Cost, Price, AllergyAdvice, InStock, Historic. This information will then be used to allow only the correct meals can be ordered during service and help keep a record of what’s selling.
5. The system must store detailed employee information which includes; Employee ID, EmployeeFirstName, EmployeeSurnameName, PhoneNumber, Email, Address, Job, PayRate, EmergencyContactName, EmergencyContactNumber. These will be used to contact employees for information regarding shifts and other stuff. It also has emergency contact information in case of an emergency.
6. All the orders need to be entered in two files; these will include Order ID, Meal ID, Quantity, TableNumber, Date, and EmployeeName.
7. The system will also need to be able to print out complex reports showing how well various meals are selling.
8. The system must run on the current hardware

## Realistic appraisal of the feasibility of potential solutions

The first possible solution is a manual one; this would be similar to the current system although more organised. In service things would happen very similarly, waitresses would write down orders on a piece of paper then copy it out and give one to the kitchen and keep one to work out the receipt later. However to get a useful amount of information from this the waiting staff would also have a book with all the meal names in a row and the rest of the rows as dates they would tally up how many of each meals were sold per service. Also another book would be needed to write down the price of each order and how much has come in. Address books could be used to store employee and supplier data which. This would be better than the current system but totaling how many of each meal are sold manually and having to write down all this extra information would take quite a long time, this would be especially inconvenient when very busy and these steps would likely be forgotten. Also this system isn’t intuitive and would have to be taught and reminded on various occasions. Finally the address books themselves would be a good way to store the data however these can easily be picked up and misplace, so in an emergency when contact information is needed fast the book may be hard to locate.

The second possible solution is building a database using a pre-built piece of

Software like Microsoft Access. This would be better than a paper based solutions as it would take up much less physical space and remove the likelihood or misplacing or losing any of the files. It would also be a lot easier and quicker to input and find the records.

The Final possible solution is building a bespoke coded system. This would be similar to the Microsoft access solution but include more necessary information and be possible to create a system, which encompasses all aspects of the system into one. This system will also be more intuitive and easy to use with no learning curve like with access it will all be very simple and only what you need will be what you see. This system will also allow us to create a system which works best with the current hardware, as the computer has relatively low ram and processing power a bespoke system will use much less than access as only what is needed will be running.

In building this bespoke system my first discussion with the client we talked about the idea of storing supplier data could be stored and how we could use this to keep an automatic stock count. However after more thought and discussion with a user (head chef) I realized this was an unrealistic plan. It would be too complicated to keep a running stock count as there’s no way to track exactly how much rice for example is used. With more time I would try to add functionality to store supplier info which would allow the user to create shopping lists and send emails direct to suppliers. However in the current time parameter this part of the system won’t be created.

## Justification of chosen Solution

I chose to build a bespoke coded solution, as this is the one, which will allow me to tick all the boxes set out in my general and specific objective. It will allow me to truly make a unique system, which is most importantly easy to use and almost instinctively intuitive. It will allow me to create many files and use all the data from these files to create useful and informative reports which can be displayed both on screen and printed. Finally it’s the only system which can make the ordering system streamlined, the other two solutions would still need to write out the orders twice and have to work out the total cost of meals, but with a coded system you’ll be able to choose only meals which are available and instantly have a total for the total cost as well as adding information which can be used later for analysis in complex reports.

## Evidence of use of appropriate analysis techniques

### Interviews

Interview with Mr. Mundey (Head chef and user of the system)

**What is the current system during service for orders?**  
The waitresses get the orders from customers and write it down on a piece of paper which they bring into the kitchen and give to one of the chefs, we then cook it and ring a bell to get a waitress to come in a take the food.

**How can this system cause problems?**  
Well we often misread what’s written, especially with quantities. The only other problem I can think of is when we run out of something and we tell one of the waitresses, if they don’t tell each other (or if they forget) then we can get food ordered which we don’t have.

**Are there any other aspects you think would benefit from a new system?**  
I think keeping track on what meals are selling most and which suppliers supply what meals would be very useful. Also if the waitresses had a menu or list showing allergy advice that would be useful to stop waitresses coming in and asking every time someone has a special dietary need.

**Finally how many meals do you have on the menu and how many order and meal do you do per day?**

We currently have 10 meals on each menu; we have 5 menus at the moment. I would like it if the new system allows me to keep meals in the system for a long time as we often remove and then bring back different meals. As far as orders go it varies massively per day, probably on average we seat around 20 tables and these generally have around 6 meals per table.

Interview with Mr. Moon (Manager, client and user of the system)

**What is the current system you use for keeping supplier and employee data?**

It’s a bit unorganised really; Darren [the head chef] stores all the kitchen staffs’ names and phone numbers in his phone and the same with Chelsea [head front of house] in her phone. There really isn’t a central database or record of all the employees who work here.  
**How do you keep track of what is sold when?**

Well in the kitchen the chefs add up the total number of mains orders at the end of each service and write it on a big wall planner in the kitchen. This helps us so we know what days get busiest like Christmas, Mother’s day or Easter. But that’s about all the information I get, we can gauge what individual meals are selling by how much we restock them but that’s a very inaccurate guide.  
**Would more information be useful to you?**

Yes I believe it would. If we had the ability to quickly tailor the menu to what’s selling and what’s making the most profit, then that would be great. Also we could see what days we need to employ more staff on.  
**So what information and calculations would you like this system to do?**

I’d like a detailed run down of each Meal and what’s selling as well as a way to order meals by most popular and days by most popular. Some information on the profits of each item would also be very useful. **We mentioned the employee and supplier data earlier would you like this protected?**Yes I think it would be wrong to leave this data open to everyone, also I would like only a select few to be able to see the monetary data. The rest only need access to the order form.  
**What is the current hardware you have?**

Oh I’m not entirely sure, we currently don’t use any computers during service apart from the till. But we do have an old windows PC an A4 printer that is occasionally used for the accounts, which you could use. [He showed me exactly what it was after and it’s a DELLPRECICION 380 running windows XP]

**What would the budget be for new hardware?**

Well I wouldn’t want to be spending much more than £150

**Does your company have any formatting that you’d like to carry on through this (eg. colour scheme or font)?**

Well we use dark Blue and Black a lot in our promotional adverts and is the colour of our pub so definitely blue would be a nice touch.

**How many suppliers and employees do you need stored in the system?**

Well at the moment we use 10 different suppliers and have around 30 staff working for us, these numbers both fluctuate quite a lot but we’ve never had more than 50 for either and I never want to.

Interview with Ms. Fitzgerald (head waitress and user)

**Can you explain to me how service works from the waitresses point of view?**  
Well We Seat customers at a table, we then give them a menu offer drinks and then give them some time to think. Once they’re ready we take their orders writing it down on a pad, along with the table number and then we write it on a second piece of paper and take that one to the kitchen. Once the meal is ready we take the food to the customers and once they’re finished and wish to pay the bill we look through our pad and find the things they’ve ordered and then you have to work out how much it costs all together, they then pay and we put the money in the till.

**Do you know any problems with the current system?**  
Yeas there are a few, some of the waiting staff aren’t very good at math’s and even ones who are during service when it’s busy it’s very easy to make a little mistake a charge people too much or too little. Another problem we have is when waiting staff don’t know what is and isn’t in stock, they then put through an order which the kitchen can’t make which means we have to go back to the table and make them reorder.

(Bad math’s bad handwriting)

**Would you be comfortable with a new computer based system?**  
I would think so, I’m not a computer whiz but I know how to use one for basic stuff, so as long as it’s not too complicated I think it should speed up service a lot and be generally be very useful.

**Are there any features you think would benefit the front of house staff?**  
Just having the system so we can select the meals and how many are needed and then the receipts and order checks are automatically worked out and printed.

### Questionnaire

Questionnaire for staff and future users of the system

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Job description | Waiter | Waitress | Chef | Waiter | Waiter | Waitress |
| Please rate your computer skills out of 5 | 4 | 3 | 2 | 3 | 2 | 4 |
| Do you think there is need for a new system? Y/N | Y | Y | Y | N | Y | Y |

### Source Documents Analysis



|  |  |
| --- | --- |
| Document Record | |
| Name of Document | Sports Bar Menu |
| Document Number | 1 |
| Size | A4 Sheet folded in three sections |
| Prepared By | This is prepared on a computer by the head chef and 20 were printed. Menus are rarely changed so these documents are rarely made. |
| Purpose | The purpose of this document is to inform customers of the meals sold and the price of them. I’ve included this as a source document even though it won’t be used directly as part of my system but as an example of the necessity of menus, prices and meal names. |

|  |  |
| --- | --- |
| Macintosh HD:private:var:folders:8y:5wz_w1357rz7k2d241xw0m6w0000gp:T:TemporaryItems:20141020_181449.jpg | |
| Document Number | 2 |
| Document Name | Order Slip |
| Other Names or References |  |
| Size | A6 |
| Number Of Parts | 1 (occasionally more if large table) |
| Medium | Paper |
| Prepared By? | Waiting Staff |
| How Often Prepared? | For every table served |
| How Many? | About 40 per day |
| Filed Where? | Thrown away at the end of the night (after number of meals have been counted) |
| In What Order? | N/A |
| How Long Kept? | N/A |
| Who Gets It? | Given to the chefs |
| For What Purpose? | To tell the chefs what to cook |

|  |  |
| --- | --- |
| Macintosh HD:private:var:folders:8y:5wz_w1357rz7k2d241xw0m6w0000gp:T:TemporaryItems:20141020_181529.jpg | |
| Document Number | 3 |
| Document Name | Receipt |
| Other Names or References | Bill |
| Size | A6 |
| Number Of Parts | 2 (two copies are made, one for the customer and one for the pub to keep) |
| Medium | Paper |
| Prepared By? | Waiting Staff |
| How Often Prepared? | For every table served |
| How Many? | About 40 per day |
| Filed Where? | One is given to the customer for their own record (however they want to keep or throw them away) and all the day’s receipts are bound together with an elastic band and kept in a filing cabinet. |
| In What Order? | Date order with the most recent first |
| How Long Kept? | 2 months |
| Who Gets It? |  |
| For What Purpose? | To keep a recent record of what meals have been sold. |

|  |  |
| --- | --- |
| Macintosh HD:private:var:folders:8y:5wz_w1357rz7k2d241xw0m6w0000gp:T:TemporaryItems:20141020_181529.jpg | |
| Document Number | 3 |
| Document Name | Receipt |
| Other Names or References | Bill |
| Size | A6 |
| Number Of Parts | 2 (two copies are made, one for the customer and one for the pub to keep) |
| Medium | Paper |
| Prepared By? | Waiting Staff |
| How Often Prepared? | For every table served |
| How Many? | About 40 per day |
| Filed Where? | One is given to the customer for their own record (however they want to keep or throw them away) and all the day’s receipts are bound together with an elastic band and kept in a filing cabinet. |
| In What Order? | Date order with the most recent first |
| How Long Kept? | 2 months |
| Who Gets It? |  |
| For What Purpose? | To keep a recent record of what meals have been sold. |

|  |  |
| --- | --- |
| Macintosh HD:Users:josephlaithwaite:Desktop:MONDAYSlovesunday (dragged).pdf | |
| Document Number | 4 |
| Document Name | Sunday Menu |
| Other Names or References |  |
| Size | A4 |
| Number Of Parts | 1 |
| Medium | Paper |
| Prepared By? | Head chef and owner |
| How Often Prepared? | Roughly once a year, the as the menu isn’t changed very often. |
| How Many? | 50 are printed to begin with (however more are printed when the current ones are used up) |
| Filed Where? | Kept on the bar |
| In What Order? | N/A |
| How Long Kept? | As long as this menu is used |
| Who Gets It? | The customers |
| For What Purpose? | To choose what to order |

|  |  |
| --- | --- |
| Macintosh HD:Users:josephlaithwaite:Downloads:eastneytavernMENU-copy-2.pdf | |
| Document Number | 5 |
| Document Name | Traditional menu |
| Other Names or References |  |
| Size | A3 |
| Number Of Parts | 1(double sided) |
| Medium | Paper |
| Prepared By? | Head chef and owner |
| How Often Prepared? | Roughly once a year, the as the menu isn’t changed very often. |
| How Many? | 50 are printed to begin with (however more are printed when the current ones are used up) |
| Filed Where? | Kept on the bar |
| In What Order? | N/A |
| How Long Kept? | As long as this menu is used |
| Who Gets It? | The customers |
| For What Purpose? | To choose what to order |

# Design

## Overall system design



## Modular structure of system

## Design Data Dictionary

All these files have used random record ordering and mostly processed serially, however searches will be used which process data randomly. These will all be stored on the main hard disk of the computer, this is because it’s a non-volatile storage medium, which can hold a lot of data very cheaply.

**Meal File**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data | Validation |
| Meal-ID | Primary | Unique meal identifier | Short | 1 – 300 | 135 | Generated by the computer, whole number in the range. |
| MealName | ––– | The name of the meal | String | ––– | Fish and Chips | No longer than 20 characters with no numbers. |
| Menu | ––– | A way to split the all meals into sub categories, such as starters, desserts, Sunday lunch, breakfast etc. | String | ––– | Mains | No longer than 15 characters with no numbers. |
| Cost | ––– | The cost of buying in a meal | Single | £1.00 – £30.00 | £3.50 | Number in range with at most 2 decimal places. |
| Price | ––– | The price the meal is sold for | Single | £5.00 – £40.00 | £8.00 | Number in range with at most 2 decimal places. |
| Allergy Advice | ––– | Information on allergy advice | String | ––– | Gluten | Less than 30 characters long with no numeric data |
| InStock | ––– | A Boolean variable used so the waitresses can’t order an item which isn’t in stock | Boolean | Yes–No | Yes | Either Yes or No. |
| Historic | ––– | A Boolean variable to remove a meal from the service choices but keep information so the system can be used over time. | Boolean | Yes–No | No | Either Yes or No. |

**Employee File**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data | Validation |
| Employee -ID | Primary | Unique ID for each employee | Short | 0-50 | 32 | Generated by the computer, whole number in the range. |
| Employee FirstName | ––– | The first name of the employee | String | ––– | Lucy | No longer than 15 characters long, with no numeric data |
| Employee SurnameName | ––– | Surname of the employee | String | ––– | Biggins | No longer than 15 characters long, with no numeric data |
| Phone Number | ––– | The employees contact phone number | String | ––– | 077 9282 6246 | 11 characters long all numeric. |
| Email | ––– | The employees email address | String | ––– | Lucy. Biggins @live.co.uk | 20 Characters with an alpha numeric data. Has to be in a format with @ in the middle |
| Address | ––– | The employees home address | String | ––– | 273 Jesse Road, Portsmouth | 30 Characters with any alpha numeric data |
| Job | ––– | The Job title of the employee | String | ––– | Waitress | 10 character alphabetical string |
| PayRate | ––– | The Hourly rate of pay for the employee | Single | £3.79 – £10.00 | £6.50 | Number with up to 2 decimal places with in the range |
| Emergency ContactName | ––– | The employees emergency contacts name | String | ––– | Janet Biggins | Maximum 30 characters long |
| Emergency ContactNumber | ––– | The employees emergency contacts phone number | Short | ––– | 077 8257 7828 | Up to 11 characters only numbers. |
| Password | ––– | This will act as a security measure to stop unauthorized people accessing the personal and financial information. | String | ––– | QwErTy321&\* | Up to 15 Alpha-numeric characters long |

**Order File**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data | Validation |
| Order-ID | Primary | A unique identifier for each order | Integer | 0 – 51,100 | 563 | Generated by the computer, whole number in the range. |
| Table Number | ––– | The number of each table, first number represents section second two number | Short | 101-410 | 201 | Integer in the range |
| Date | ––– | The date on the day of the order | Date | Any Date | 13/09/2014 | Must be in the Format DD/MM/YYYY |
| Employee ID | Foreign Key | The Employee ID of the waitress | Short | 0-50 | 25 | Generated by the computer, whole number in the range. |

**OrderItem File**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data | Validation |
| Order-ID | Foreign | A unique identifier for each order. | Integer | 0-51,100 | 563 | Generated by the computer, whole number in the range. |
| Meal-ID | Foreign | Unique meal identifier. | Short | 0-300 | 135 | Generated by the computer, whole number in the range. |
| Quantity | ––– | The number of one item which is ordered. | Short | 1-20 | 3 | Any integer in the range |
| Price | ––– | The current price the meal is sold for. | Single | £1.00-£30.0 | £8.00 | Any numeric data with at most 2 decimal places within the range |
| Cost | ––– | The current price to by the order in. | Single | £5.00-£40.00 | £3.50 | Any numeric data with at most 2 decimal places within the range |

**Menu File**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Name | Key | Description | Data Type | Range | Sample Data | Validation |
| Menu Name | --- | The name of a menu | String | --- | Dessert | Any alphabetic text with no more than 15 characters |

## Algorithms for data transformations

### Login

This algorithm will be on the login form and used to validate the employeeID entered. It checks to see whether the user is logging in as an administrator or as a normal user. It populates two global variables, the first CurrentUserAdministrator this is used by the order forma and when it is false a logout but will appear instead of a menu button. The second variable CurrentEmployeeID is also used by the order form and used to get the employee name when opening new orders.

If password input box is empty

Set CurrentUserAdministrator as False

If userID is empty then

Display error message

Else

Loop through every employee record

If employee record ID equals inputted ID

Set ValidID as True

If validID is true

Hide this window and open orders form

Set CurrentEmployeeID as Inputted ID

Else

Display Error message

Else

If EmployeeID input box is empty

Display error message

Loop through every employee record

If employee record ID equals inputted ID

Set ValidID as True

If ValidID is true

Set CurrentUserAdministrator as true

Set CurrentEmployeeID as Inputted ID

Hide this window

Open menu form

### Saving Data (employee and meal forms)

This algorithm will be used on both the employee form and the meal form to validate and save the data. Here I haven’t specified every single variable that will be checked just an example of how different variables will be checked for what.

Set message as nothing

If input box equals nothing

Message = Message and “The input box is empty” and line

If input box data is longer than the size of the variable

Message = Message and “The data in the input box is too long” and line

If number input box is not a number

Message = Message and “This input box should be a number” and line

If Message is empty

Save record to file

Save highest ID to the ID file

Else

Display error message containing message

### Meal selection

This algorithm will be used on the meal form and the order form, it is a way to quickly find and select the meal you want removing all possibilities of incorrectly entered data and ambiguities.

Load menu dropdown box (procedure)

Open file menu file

For every line in the file

Write the contents as an item in the dropdown box

Menu dropdown box click event (Procedure)

With every meal record

If its menu is the same as the one selected

Paste the meal name in the meal dropdown

### Search Employee file

This algorithm is used to search through the employee file by either employee ID or by first, last or full name. It will be used on the employee form to make it easy to find an employee

If Search box text is a number

For every employee record

If employee ID equals searched ID

Call procedure to read record contents

Else if the box isn’t empty

For every employee record

If searched name equal the first name or last name or full name

Call procedure to read record contents

Else

Error message “Please a name or ID to search and try again”

### Delete employee record

This is the algorithm, which will be used on the employee form to delete a record. It saves the deleted ID in a separate file which will be used to display unknown employee information in the reports form.

Create temporary file

For every employee record

If employee record doesn’t equal the ID to be deleted

Save the record in the temporary file

Else

Set Deleted to true

If Deleted is true

Delete employee file

Copy temporary file to employee file

Delete the temporary file

Open deleted employee ID file

Save the ID of the deleted record to the file

Else

Error message

### Reports

The first algorithm is an algorithm used to copy and collate information from the main files into important information to be displayed. The meal information will work by finding every order the meals been sold in and totaling the number of meals sold along with things like profit, money taken and popularity

For every meal record

Set NumberOfMeals to zero

Set MoneyTaken to zero

Set Profit to zero

For every order file

If order date is within the inputted dates

For every order item in this order

NumberOfMeals 🡨 NumberOfMeals and quantity of order item

MoneyTaken 🡨 MoneyTaken and Price \* Quantity

Profit 🡨 Profit and (Price - Cost) \* Quantity

If MealRecord is historic and NumberOfMeals is zero

Set meal name to nothing

Else

MealName is the name from the record

Save NumberOfMeals, MoneyTaken, Profit and MealName to the array

There will then be other buttons, which appear, and pressing each of these will order the list in a different way.

Order alphabetically

For every meal record +1 index1

For every meal record index2

If meal in Position index1 in array > Meal in Position index2 in array

Temp 🡨 Meal array in position index1

Meal array position index1 🡨 Meal array position index2

Meal array position index2 🡨 Temp

Call procedure to display the array

### Orders

This is the algorithm, which will be used on the orders form to firstly get the employee name from their login detail, date and order ID. Check this data is valid and then save this both to the order file and the OrderItem file.

Date 🡨 todays date

Open ID file and get OrderID Record

OrderID 🡨 OrderID from file + 1

For every Employee Record

If Current employee ID = records ID

Name 🡨 Employee from records first name

If TableNumber input box is nothing

Message 🡨 Message and “Enter a table number”

Else If Table number isn’t a number

Message 🡨 Message and “The table number must be a number”

Else

For every Open order

If open order table number is inputted

For every order record

If record hasn’t been cleared and is the same table number as the inputted table number and doesn’t have the same order ID

Error message asking if they want to add a new table number or add it to the existing order on that t able

If no errors

For every order record

If record order ID equal inputted Order ID

OrderSaved = True

If OrderSaved is false

Save inputted data (excluding meals) to order file

For every meal record

If meal name equals inputted meal name

MealID 🡨 MealID from record

Price 🡨PriceCurrent from record

Cost 🡨 CostCurrent from record

Save meal data to OrderItem file

## User interface design (HCI) rationale

The human computer interface will be designed with the following aspects in mind to create the perfect end user experience.

The place this system will be used is a pub so, although it’s a business, it doesn’t have to be too formal and I can incorporate more colour and a friendlier user interface into my design.

The Users of the system vary massively from some people at university who are very computer literate to older employees who aren’t that used to modern technology. I will have to think about this when creating my interface as the program must cater to the least computer savvy employee so the user interface must be simple and intuitive.

The users’ needs have been considered and I will design my interface to include all the features my client and users need.

The System will be based on a computer with two input devices, a keyboard and a mouse, so my interface is designed with this in mind and will be navigated using a mouse and data will be entered using the keyboard. There will also be a few shortcuts so you people confident with the system can use it more efficiently using tab and enter keys instead of the mouse. The computer will also have three printers, one A4 printer for financial and performance reports and two receipt printers.

Dialogues, a variety of dialogues will be used most data will be entered using text boxes, this will be clearly labeled and it should be obvious the user should enter data. I will also be using combo boxes which will allow only valid data to be entered, this will reduce the need for validation checks and make the system easier to use, especially those who have poor spelling. Finally labels will also be used, these will be slightly darkened and it will be clear users do not enter data here, however data for these will be added by the computer.

Colours, in my interview with the client he specified two colours commonly used, black and blue, I will use these colours to create a more engaging interface that really fits in with the pub.

I will not be using icons as I believe one word of text in a button will convey enough information for the user to use the system instinctively, I believe icons can sometimes be misleading especially when used to describe more abstract buttons. I will also not be using many 3d effects as I think a flat design is simpler to use and more aesthetically pleasing.

The system should also include helpful error messages and feedback on screen, these will help the user enter only valid data and guide the user through the system, and these will help new users learn to use the system quicker than otherwise possible. Also I will be providing o-mail help so if people have problems they can directly ask me, however this should be at a last resort after checking the electronic user manual.

I will also make sure that every page has clear navigation so that exits will always be obvious and getting out of the system in a safe way, saving all data.

## UI sample of planned data capture and entry design

**Key**

Computer Entered label

Text Box

Combo box

**Login Screen**

Employee I.D

Password

Enter

Exit

Eastney Tavern - Login

**Menu Screen**

Option 1

Eastney Tavern - Menu

Orders

Meals

Employees

Reports

Settings/Help

Log Out

Exit

Option 2

Eastney Tavern - Menu

Orders

Meals

Employees

Reports

Log Out

[PICTURE]

Exit

Settings/Help

Option 3

Eastney Tavern - Menu

Orders

Meals

Employees

Reports

Log Out

[PICTURE]

Exit

Settings/Help

**Order Screen**

Option 1

Eastney Tavern - Orders

Open Orders

New Order

Order ID

Date

Main Menu

Table Number

Employee Name

Menu

Meal

Quantity

Add

Kitchen Order

Customer Receipt

Print Order

Print Receipt

Add Meal

Option 2

Eastney Tavern - Orders

Open Orders

New Order

Order ID

Date

Employee Name

Table Number

Main Menu

Menu

Meal

Quantity

Add

Kitchen Order

Customer Receipt

Print Order

Print Receipt

**Meals Screen**

Eastney Tavern - Meal

Menu

New Meal

Meal ID

Meal Name

Menu

Main Menu

Meals

Meal

Allergy Advice

Price

Cost

In stock

Historic

✓

✓

Next Record

Last Record

Delete Record

[List Box]

**Employee Screen**

Eastney Tavern - Employees

New Employee

Employee Name

Main Menu

Employee ID

First Name

Last Name

Job Tittle

Pay Rate

Employees

Phone Number

Email

Address

E.C. Name

E.C. Phone number

Next Record

Last Record

Delete Record

[List Box]

**Reports Screen**

Eastney Tavern - Reports

[List Box]

Employee Info

Employee Performance

Meals Info

Profits and losses

From Date

To Date

Create

Meals By Popularity

From Date

To Date

Create

Option 1

Option 2

Eastney Tavern - Reports

[List Box]

Employee Info

Employee Performance

Meals Info

Profits and losses

From Date

To Date

Create

Meals By Popularity

From Date

To Date

Create

**Settings/Help**

Eastney Tavern – Settings/Help

Main Menu

[List Box]

File Name

Delete

Help Section

Help File

Add menu

Add

Delete menu

Delete

## UI sample of planned valid output designs

**Kitchen Orders**  **Customer Receipt**

The Eastney Tavern

13/07/2014 18:32

Table No. 101

Chelsea

|  |  |  |  |
| --- | --- | --- | --- |
| Olives | 1 | £3.45 | £3.45 |
| Seafood Cocktail | 2 | £5.75 | £11.50 |
| Bread | 1 | £3.00 | £3.00 |
| Fish & Chips | 2 | £9.50 | £19.00 |
| Liver & Bacon | 1 | £9.80 | £9.80 |
| Curry | 1 | £7.00 | £7.00 |
|  |  |  |  |
|  |  | £ 53.75 | |

13/07/2014 18:32

Table No. 101

Chelsea

Olives 1

Seafood Cocktail 2

Bread 1

Fish & Chips 2

Liver & Bacon 1

Curry 1

**Meals by Popularity**

Date: 13/07/2014

Totals

Menu Quantity Profit/Loss

Starters 15 £43.00

Mains 30 £104.75

Desserts 10 £34.70

Total 55 £182.45

Starters

Meal I.D Meal Name Quantity Profit

001 Olives 3 £9.00

004 Bread 7 £14.00

007 Sea food cocktail 5 £20.00

Mains

Meal I.D Meal Name Quantity Profit/Loss

002 Liver & Bacon 18 £58.25

003 Fish & Chips 12 £46.50

Desserts

Meal I.D Meal Name Quantity Profit/Loss

005 Sticky Toffee Pud 6 £22.00

006 Chocolate Pud 4 £12.70

**Employee Info**

I.D Name Phone No. Email Address Job Pay EC Name EC Number

I.D Name Phone No. Email Address Job Pay EC Name EC Number

I.D Name Phone No. Email Address Job Pay EC Name EC Number

I.D Name Phone No. Email Address Job Pay EC Name EC Number

I.D Name Phone No. Email Address Job Pay EC Name EC Number

**Profits**

Date: 13/07/2014

Totals

Menu Quantity Profit

Starters 90 £480.09

Mains 303 £1440.28

Deserts 114 £483.00

Total 507 £2400.47

Date Meals Sold Profit

Starter Mains Desserts

13/07/2014 15 30 10 £356.00

14/07/2014 22 80 32 £489.27

15/07/2014 17 66 27 £667.30

16/07/2014 36 127 45 £887.90

## Description of measures planned for security and integrity of data

The data stored on this system is very important, not only as it contains a lot of personal information about the employees but it also will act as proof of income, which is a legal necessity for taxes. This means backups of the information will have to be created.

This will be done in two ways, the first will be by plugging in a USB disk and copying the files over, this should be done every week and stored in a secure locked place. Another system, which would be useful in case of a fire of natural damage, would be to use a cloud storage solution such as Dropbox or Google Drive, I would then use a backup program such as sync toy which could copy all the files daily to the cloud storage. This would be very useful in case of damage to property as the important information will be safely stored offsite in the cloud.

## Description of measures planned for system security

All employees will have a password and they will use this along with their assigned employee I.D to gain access to the system, which includes the private and financial information. Employees will however be able to enter the ordering screen without using a password; this is for ease of use and will allow employees to gain access much faster as they won’t have to key in a 15-character password every time they use the system. I’ve decided this as the boss wanted ease of use to be one of the most important factors and this is a compromise between that and security.

The system will also be contain anti-virus software such as AVG this will scan for viruses and malicious software then neutralize the threat if one is found. I will also use a firewall to reduce the number of open ports, this is, as the program doesn’t actually need the Internet very little Internet access will be used, this should help reduce the likelihood of attacks.

Finally I will be suggesting to my client to update to a newer version of windows. I will do this as the current operating system is windows X.P and support from windows has now been removed, thus less security systems are updating their software.

## Overall test strategy

During the production and testing of my system I plan to adopt an array of testing techniques which will help me to create the system module by module and hopefully by the end of my testing the system will be tested to check expected outputs are sent out and erroneous data are removed before being saved. The test strategies I will employ are bellow.

### White Box testing

This test strategy is designed using knowledge of the code so tests are designed to check every possible logical route through the code. I will use this in tests when the user inputs erroneous or extreme data to test no matter what is entered only valid data is saved.

### Black Box testing

These tests are the opposite of white box tests, they are designed with no thought about the code and how the data is transferred only as the end user will use the system. Therefore only typical data is used in these tests and they are used to check the outputted data is what you were expecting for the inputted data. I will draw on this testing for the reports form checking outputs are correct.

### Unit testing

This is to test each module separately, as you build it. For example I’ll create a data entry form and test this for validates and saves the data correctly before building more modules that use this data.

### Integration testing

These are the tests used once after unit tests have been carried out on interlinking modules. I will test two modules together then add another and this will show me if there are any problems with how the modules communicate with each other.

## Preliminary test plan & Test data

### Meal input screen

**Test Data**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Meal-ID | Meal Name | Menu | Cost | Price | Allergy Advice | In Stock | Historic |
| 1 | Satay Chicken Skewer | Starters | £0.80 | £3.95 | Egg & Nut | True | False |
| 2 | Fresh Seafood Cocktail | Starters | £1.10 | £4.95 | Egg, seafood | True | False |
| 3 | Seared Scallops Chorizo | Starters | £1.05 | £4.95 |  | True | False |
| 4 | Rib eye Steak | Traditional | £6.70 | £17.45 | Gluten Dairy | True | False |
| 5 | Portuguese Espetada | Traditional | £7.00 | £17.45 |  | True | False |
| 6 | Lamb Tagine | Traditional | £3.80 | £12.95 |  | True | False |
| 7 | Minted Lamb Burger | Traditional | £3.65 | £11.95 | Dairy  Gluten | True | False |
| 8 | Chips | Side Dishes | £0.30 | £2.45 | Gluten | True | False |
| 9 | Garlic Bread | Side Dishes | £0.30 | £2.45 | Gluten | True | False |
| 10 | Vegetables | Side Dishes | £0.20 | £1.95 |  | True | False |

**Test Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Test data | Test type | Test description | Expected outcome | Actual outcome | Corrections |
| 1 | Record one from Meal file test data | Normal | Enter a normal selection of data and chrck it saves and displays correctly | The record willl be saved and displayed in the list box |  |  |
| 2 | Record 2 from meal test data | Extreme | Enter a meal with a meal name longer than 20 characters | An error message shoud come up nd the record won’t be saved. |  |  |
| 3 | Record 3 with no name | Erroneous | Enter a record but without a meal name. | Error message telling me to enter a name and the record not being saved |  |  |

### Employee Input screen

**Test Data**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | First name | Surname | Phone number | Email | Address | Job | Pay rate | EC Number | EC Name | Password |
| 1 | Chris | Moon | 07756827315 | Chris.Moon@outlook.com | 100 Cromwell Road | Owner | £00.00 | 0270015628 | Jax | Jet227 |
| 2 | Luke | Lindsey | 02392786651 | Luke.Lindsey@live.com | 17 Lance road | Chef | £7.80 |  | Mark |  |
| 3 | Darren | Munday | 07753210865 | Munday82@hotmail.co.uk | 222 Victoria grove | Head Chef | £12.80 | 0238768321 | Sarah | Cassarole3 |
| 4 | James | Legget | 07785921623 | James125@hotmail.com | 127 Short lane | Waiter | £6.60 | 0778562918721 | Lucy |  |
| 5 | Mark | Houghton | 07758632387 | HilseaHoughton@live.com | 32 Liverpool Boulevard | Waiter | £7.10 | 0239278641 | Janet |  |

**Test Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Test data | Test type | Test description | Expected outcome | Actual outcome | Corrections |
| 1 | 1st Employee test data | Normal | Employees information is saved correctly | All inputs saved and displayed in the list box |  |  |
| 2 | 2nd row of employee data Without entering the first name | Erroneous | Testing that checks and finds the name is missing | A message box will come up telling you the first name isn’t entered and the record hasn’t been saved |  |  |
| 3 | Full 3rd row of employee data | Normal | Test the system saves an employee correctly who isn’t an administrator (doesn’t have a password) | The employee will be saved after a message box appears checking I don’t want him to be an administrator |  |  |
| 4 | Record 4 from the employee test data, but with extra numbers 1 after the employee phone number | Extreme | Test to check phone number is not too long | A error message will pop up and tell you the phone numer is too long, and the record won’t be saved |  |  |

### Orders

**Test Data**

|  |  |  |  |
| --- | --- | --- | --- |
| Order-ID | Table Number | Date | Employee ID |
| 1 | 102 | 07/02/2014 | 1 |
| 2 | 204 | 15/03/2014 | 5 |
| 3 | 101 | 23/05/2014 | 4 |
| 4 | 301 | 24/05/2014 | 5 |
| 5 | 103 | 15/07/2014 | 1 |
| 6 | 210 | 21/09/2014 | 3 |
| 7 | 201 | 09/11/2014 | 6 |
| 8 | 302 | 15/01/2015 | 6 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order-ID | Meal-ID | Quantity | Price | Cost |
| 1 | 1 | 1 | £3.95 | £0.80 |
| 1 | 3 | 1 | £4.95 | £1.05 |
| 1 | 5 | 1 | £17.45 | £7.00 |
| 1 | 7 | 1 | £11.95 | £3.65 |
| 2 | 4 | 1 | £17.45 | £6.70 |
| 2 | 6 | 2 | £12.95 | £3.80 |
| 3 | 8 | 1 | £2.45 | £0.30 |
| 4 | 4 | 1 | £17.45 | £6.70 |
| 4 | 5 | 3 | £17.45 | £7.00 |
| 4 | 10 | 1 | £1.95 | £0.20 |

**Test Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Test data | Test type | Test description | Expected outcome | Actual outcome | Notes |
| 1 | Order file 1 and order item file 1-4 | Normal | Enter a normal tables order. Login with the correct employee ID, | The data is saved and is displayed in the order and receipt. Also the total is found which should be £38.30 |  |  |
| 2 | OrderID 2  But with quantity of 15 for the chips. | Extreme | Entering correct data but with a very large quantity | An error message will appear checking the quantity is correct |  |  |

### Reports

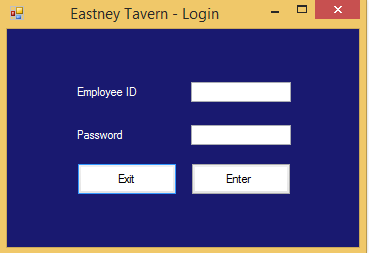
Test data is all the test data already entered

**Test Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Test data | Test type | Test description | Expected outcome | Actual outcome | Corrections |
| 1 | Employee IDs 1-5 OrderIDs 1-2 & 4-6  MealIDs 1 & 1-10. All ordered dates | process | Check data entered is what is given by the form. | Data to be tottalled with 18 meals, £134.95 profit and £207.6 Money taken | As expected | None needed |
| 2 | All Inputed info  Search name “mark” | Process | Search with on lowercase name | Mark Houghton will be displayed in the list box | As expected | No correction required |
| 3 | Search Name “Bob” | Erroneous | Search incorrect name | The employee won’t be found and an error will be shown |  |  |

# Technical Solution

## Login Form



Public Class frmLogin

Private Sub frmLogin\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Dim Index As Integer

FileOpen(MealFileNumber, MealFilePath, OpenMode.Random, , , Len(MealRecord)) 'Open all files

FileOpen(OrderFileNumber, OrderFilePath, OpenMode.Random, , , Len(OrderRecord))

FileOpen(OrderItemFileNumber, OrderItemFilePath, OpenMode.Random, , , Len(OrderItemRecord))

FileOpen(EmployeeFileNumber, EmployeeFilePath, OpenMode.Random, , , Len(EmployeeRecord))

FileOpen(IDFileNumber, IDFilePath, OpenMode.Random, , , Len(IDRecord))

If LOF(IDFileNumber) / Len(IDRecord) = 0 Then 'Set up ID file

For Index = 1 To IDFileNumber

IDRecord.FileNumber = Index

IDRecord.ID = 0

FilePut(IDFileNumber, IDRecord, Index)

Next

End If

If LOF(EmployeeFileNumber) / Len(EmployeeRecord) = 0 Then

FirstUse = True

frmEmployees.Show()

End If

End Sub

Private Sub btnEnter\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEnter.Click

Dim EmployeeRecordNumber As Short

If txtPassword.Text = "" Then

CurrentUserAdministrator = False

If txtEmployeeID.Text = "" Then

MsgBox("Please enter an employee ID (And password if you're an asministration)")

ElseIf IsNumeric(txtEmployeeID.Text) = False Then

MsgBox("Please enter a valid employee ID (with only numbers)")

Else

If ValidID(EmployeeRecordNumber) = True Then

CurrentEmployeeID = txtEmployeeID.Text

Me.Hide()

txtEmployeeID.Text = ""

frmOrders.Show()

Else

MsgBox("Please enter a valid employee ID")

End If

End If

Else

If txtEmployeeID.Text = "" Then

MsgBox("Please enter an employee ID")

ElseIf ValidID(EmployeeRecordNumber) = True Then

If ValidPassword(EmployeeRecordNumber) = True Then

Me.Hide()

CurrentUserAdministrator = True

CurrentEmployeeID = txtEmployeeID.Text

frmMenu.Show()

Else

MsgBox("The Password you entered is invalid, please try again")

End If

End If

End If

End Sub

Function ValidID(ByRef EmployeeRecordNumber) As Boolean

Dim Index As Integer

For Index = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord) 'Loop through the employee file to check the ID exists.

FileGet(EmployeeFileNumber, EmployeeRecord, Index)

If EmployeeRecord.EmployeeID = txtEmployeeID.Text Then

EmployeeRecordNumber = EmployeeRecord.RecordNumber

Return True

End If

Next Index

End Function

Function ValidPassword(ByVal EmployeeRecordNumber) As Boolean

FileGet(EmployeeFileNumber, EmployeeRecord, EmployeeRecordNumber) 'Open the record of the ID

If RTrim(EmployeeRecord.Password) = txtPassword.Text Then 'Compare the ID's password with what's writen.

Return True

End If

End Function

Private Sub btnExit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnExit.Click

End 'Quits the program

End Sub

Private Sub btnHack\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnHack.Click

Me.Hide()

frmMenu.Show()

CurrentUserAdministrator = True

End Sub

End Class

## Menu Form



Public Class frmMenu

Private Sub btnOrders\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOrders.Click

Me.Hide()

frmOrders.Show()

End Sub

Private Sub btnMeals\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMeals.Click

Me.Hide()

frmMeal.Show()

End Sub

Private Sub btnEmployees\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEmployees.Click

Me.Hide()

frmEmployees.Show()

End Sub

Private Sub btnReports\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnReports.Click

Me.Hide()

frmReports.Show()

End Sub

Private Sub btnSettingsHelp\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSettingsHelp.Click

Me.Hide()

frmSettingsHelp.Show()

End Sub

Private Sub btnLogOut\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnLogOut.Click

Me.Close()

frmLogin.Show()

End Sub

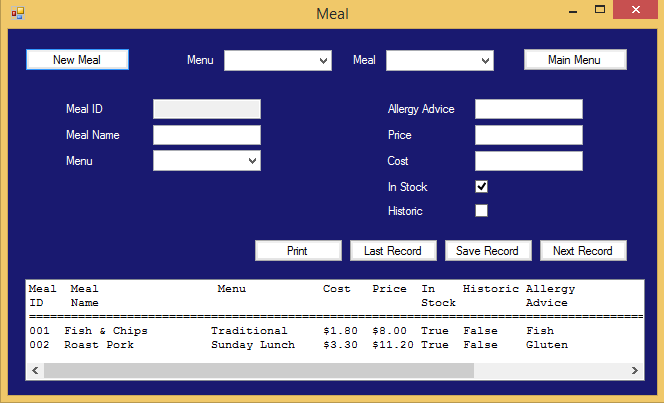
Private Sub btnExit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnExit.Click

End

End Sub

End Class

## Meal Form



Public Class frmMeal

Private Sub frmMeal\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

CreateReport() 'Call procedure to populate list box with data

LoadMenuCombo() 'Call procedure to fill combo box with menu names.

End Sub

Private Sub LoadMenuCombo()

Dim MenuName As String

FileOpen(MenuFileNumber, MenuFilePath, OpenMode.Input) 'Open Menu file

cmbMenuEnter.Items.Clear() 'Clear Combo box

Do While Not EOF(MenuFileNumber) 'Loop through menu file

MenuName = LineInput(MenuFileNumber) 'Read every line

cmbMenuEnter.Items.Add(MenuName) 'Enter them in the combo box

cmbMenuSelector.Items.Add(MenuName)

Loop

FileClose(MenuFileNumber) 'Close menu file

End Sub

Private Sub btnMainMenu\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMainMenu.Click

Me.Close()

frmMenu.Show()

End Sub

Private Sub btnNewMeal\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnNewMeal.Click

FileGet(IDFileNumber, IDRecord, MealFileNumber) 'Get the ID record for Meals from the ID file

txtMealID.Text = IDRecord.ID + 1 'Add one to the current ID

txtMealName.Text = "" 'Clear data entry boxes

cmbMenuEnter.Text = "Select A Menu"

txtAllergyAdvice.Text = ""

txtPrice.Text = ""

txtCost.Text = ""

chkInStock.Checked = True

chkHistoric.Checked = False

txtMealName.Focus()

End Sub

Private Sub btnSaveRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSaveRecord.Click

ValidateData() 'Call procedure to validate data

CreateReport() 'Call procedure to populate list box with data

End Sub

Private Sub ValidateData()

If txtMealID.Text = "" Then 'If a Meal ID has't been created display an error.

MsgBox("You must have a Meal ID before you an save a new meal." & vbNewLine & "To do this you must click the 'New Meal' button" & vbNewLine & "This meal has not been saved.")

Else

Dim Message As String = "" 'Messages to hold errors

Dim Message2 As String = ""

If txtMealName.Text = "" Then Message = "A meal name" & vbNewLine 'Validations to check 9if data field is empty

If cmbMenuEnter.Text = "Select A Menu" Then Message = Message & "A menu" & vbNewLine

If txtPrice.Text = "" Then

Message = Message & "A Price" & vbNewLine

ElseIf IsNumeric(txtPrice.Text) = False Then 'Validation numbers are actual numbers

Message2 = Message2 & "The price is not a number" & vbNewLine

End If

If txtCost.Text = "" Then

Message = Message & "A Cost" & vbNewLine

ElseIf IsNumeric(txtCost.Text) = False Then

Message2 = Message2 & "The Cost is not a number"

End If

If Message = "" And Message2 = "" Then 'If no errors save data procedure is called

SaveData()

Else 'Display error message and how to correct when errors occur.

If Message2 = "" Then

MsgBox("Please Enter:" & vbNewLine & Message & vbNewLine & "This Meal has not been saved")

ElseIf Message = "" Then

MsgBox("Please correct these errors: " & vbNewLine & Message2 & vbNewLine & "This Meal has not been saved")

Else

MsgBox("Please Enter:" & vbNewLine & Message & vbNewLine & "And correct these errors: " & vbNewLine & Message2 & vbNewLine & "This Meal has not been saved")

End If

End If

End Sub

Private Sub SaveData()

Dim RecordNumber As Short

Dim Save As Boolean = True

FileGet(IDFileNumber, IDRecord, MealFileNumber) 'Get ID for meal file

If txtMealID.Text <= IDRecord.ID Then 'Checks if ID has already been used (signifies file being updated rather than created)

If MsgBox("Are you sure you want to update this record, the old information will be lost." , MsgBoxStyle.YesNo) = MsgB oxResult.Yes Then 'Check data should be overwritten

For index = 1 To LOF(MealFileNumber) / Len(MealRecord)

If txtMealID.Text = MealRecord.MealID Then

RecordNumber = MealRecord.RecordNumber 'Find the record number for the existing meal

End If

Next

Else

Save = False

End If

Else

RecordNumber = LOF(MealFileNumber) / Len(MealRecord) + 1 'Create a new meal ID

End If

If Save = True Then

MealRecord.RecordNumber = RecordNumber ' Enter input boxes data into meal record

MealRecord.MealID = txtMealID.Text

MealRecord.MealName = txtMealName.Text

MealRecord.Menu = cmbMenuEnter.Text

MealRecord.AllergyAdvice = txtAllergyAdvice.Text

MealRecord.PriceCurrent = txtPrice.Text

MealRecord.CostCurrent = txtCost.Text

MealRecord.InStock = chkInStock.CheckState

MealRecord.Historic = chkHistoric.CheckState

FilePut(MealFileNumber, MealRecord, RecordNumber) 'Place record in meal File

If txtMealID.Text = IDRecord.ID + 1 Then 'If it's a new record

IDRecord.FileNumber = MealFileNumber

IDRecord.ID = txtMealID.Text

FilePut(IDFileNumber, IDRecord, MealFileNumber) 'Update largest used ID

End If

End If

End Sub

Private Sub CreateReport()

Dim index As Integer

lstMeals.Items.Clear()

lstMeals.Items.Add("Meal Meal Menu Cost Price In Historic Allergy") 'Write table headings

lstMeals.Items.Add("ID Name Stock Advice")

lstMeals.Items.Add("==========================================================================================================================")

If LOF(MealFileNumber) / Len(MealRecord) = 0 Then 'If no meal files saved write a message saying this

lstMeals.Items.Add("")

lstMeals.Items.Add(" ---- No Meals ----")

Else

For index = 1 To LOF(MealFileNumber) / Len(MealRecord) 'If Meal records exist step through every record and enter in list box.

FileGet(MealFileNumber, MealRecord, index)

lstMeals.Items.Add(String.Format(MealFormat, MealRecord.MealID.ToString("D3"), MealRecord.MealName, MealRecord.Menu, Format(MealRecord.CostCurrent, "Currency"), Format(MealRecord.PriceCurrent, "Currency"), MealRecord.InStock, MealRecord.Historic, MealRecord.AllergyAdvice))

Next index

End If

End Sub

Private Sub lstMeals\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles lstMeals.SelectedIndexChanged

Dim RecordNumber As Short

RecordNumber = lstMeals.SelectedIndex - 2 'Gets record number from clicking it in the list box

If RecordNumber < 1 Then

RecordNumber = 1

End If

ReadRecord(RecordNumber) 'Pass the Record number to a read record procedure.

End Sub

Private Sub ReadRecord(ByRef RecordNumber)

If LOF(MealFileNumber) / Len(MealRecord) > 0 Then

FileGet(MealFileNumber, MealRecord, RecordNumber) 'Find the record

txtMealID.Text = MealRecord.MealID 'Get data from the record and enter it into the data boxes.

txtMealName.Text = MealRecord.MealName

cmbMenuEnter.Text = MealRecord.Menu

txtAllergyAdvice.Text = MealRecord.AllergyAdvice

txtPrice.Text = MealRecord.PriceCurrent

txtCost.Text = MealRecord.CostCurrent

chkInStock.Checked = MealRecord.InStock

chkHistoric.Checked = MealRecord.Historic

lstMeals.SelectedIndex = RecordNumber + 2 'change the selected line in the list box

End If

End Sub

Private Sub cmbMenuSelector\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbMenuSelector.SelectedIndexChanged

Dim index, NumberOfRecords As Integer

cmbMeal.Items.Clear() 'Clear combo box

NumberOfRecords = LOF(MealFileNumber) / Len(MealRecord)

For index = 1 To NumberOfRecords

FileGet(MealFileNumber, MealRecord, index) 'Get the last meal record

If RTrim(MealRecord.Menu) = cmbMenuSelector.Text Then 'If the meal has the same menu

cmbMeal.Items.Add(MealRecord.MealName) 'Add the meal to the meal combo box

End If

Next index

cmbMeal.Text = ""

End Sub

Private Sub cmbMeal\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbMeal.SelectedIndexChanged

Dim NumberOfRecords As Integer = LOF(MealFileNumber) / Len(MealRecord)

Dim RecordNumber As Integer

For index = 1 To NumberOfRecords 'Loop through every record in the meal file

FileGet(MealFileNumber, MealRecord, index)

If MealRecord.MealName = cmbMeal.Text Then

RecordNumber = MealRecord.RecordNumber 'Record number of the meal selected

End If

Next index

ReadRecord(RecordNumber) 'Pass to the Read Record procedure

End Sub

Private Sub btnNextRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnNextRecord.Click

Dim RecordNumber As Short = 0

Dim NumberOfRecords, Index As Integer

NumberOfRecords = LOF(MealFileNumber) / Len(MealRecord)

If txtMealID.Text = "" Then

RecordNumber = 1

Else

For Index = 1 To NumberOfRecords

FileGet(MealFileNumber, MealRecord, Index)

If MealRecord.MealID = txtMealID.Text Then 'gets the record number of the next record

RecordNumber = MealRecord.RecordNumber

If RecordNumber + 1 > NumberOfRecords Then

RecordNumber = NumberOfRecords

Else

RecordNumber = RecordNumber + 1

End If

End If

Next

End If

ReadRecord(RecordNumber) 'Passes the next record number to the ReadRecord procedure

End Sub

Private Sub btnLastRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnLastRecord.Click

Dim RecordNumber As Short = 0

Dim NumberOfRecords, Index As Integer

NumberOfRecords = LOF(MealFileNumber) / Len(MealRecord)

If txtMealID.Text = "" Then 'gets the record number of the next record

RecordNumber = NumberOfRecords

Else

For Index = 1 To NumberOfRecords

FileGet(MealFileNumber, MealRecord, Index)

If MealRecord.MealID = txtMealID.Text Then

RecordNumber = MealRecord.RecordNumber

If RecordNumber - 1 < 1 Then

RecordNumber = 1

Else

RecordNumber = RecordNumber - 1

End If

End If

Next

End If

ReadRecord(RecordNumber) 'Passes the next record number to the ReadRecord procedure

End Sub

Private Sub btnPrint\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPrint.Click

If (PrintDialog1.ShowDialog() = Windows.Forms.DialogResult.OK) Then 'Printing

PrintDocument1.DefaultPageSettings.Landscape = True

PrintDialog1.Document.Print()

End If

End Sub

Private Sub PrintDocument1\_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage

Dim yMargin As Integer = e.MarginBounds.Y

Dim xMargin As Integer = e.MarginBounds.X

Dim currentpageItemProgress As Integer = 0

For Each item As String In lstMeals.Items

If (currentpageItemProgress >= ItemProgress) Then

e.Graphics.DrawString(item, lstMeals.Font, New SolidBrush(lstMeals.ForeColor), xMargin, yMargin)

yMargin += lstMeals.Font.Size + 10

ItemProgress += 1

If (yMargin >= e.MarginBounds.Y + e.MarginBounds.Height And ItemProgress <= lstMeals.Items.Count) Then

e.HasMorePages = True

currentpageItemProgress = 0

End If

End If

currentpageItemProgress += 1

Next

End Sub

Private Sub PrintDocument1\_EndPrint(ByVal sender As Object, ByVal e As System.Drawing.Printing.PrintEventArgs) Handles PrintDocument1.EndPrint

ItemProgress = 0

End Sub

End Class

## Orders Form





Public Class frmOrders

Dim Total As Single

Private Sub frmOrders\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

If CurrentUserAdministrator = False Then 'If the employee logged in without using a password

btnMainMenu.Visible = False 'Removes the open main menu button and replaces it with a logout button

btnLogOut.Visible = True

Else

btnMainMenu.Visible = True 'If they logged in using a password then only the main menu buttton will be visible

btnLogOut.Visible = False

End If

MenuComboLoad() 'Call procedures to load combo boxes.

LoadOpenOrders()

End Sub

Private Sub btnMainMenu\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMainMenu.Click

Me.Close()

frmMenu.Show()

End Sub

Private Sub LoadOpenOrders()

cmbOpenOrders.Items.Clear() 'Clear combo box

Dim Index As Integer

For Index = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'Loop through every record in the order file

FileGet(OrderFileNumber, OrderRecord, Index)

If OrderRecord.OrderCleared = False Then 'If the order hasn't been paiid for then it will be writen in the open orders combo box

cmbOpenOrders.Items.Add(OrderRecord.TableNumber)

End If

Next

End Sub

Private Sub cmbOpenOrders\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbOpenOrders.SelectedIndexChanged

Dim index, index2 As Integer

cmbMeal.Items.Clear()

For index = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'Loop through order records

FileGet(OrderFileNumber, OrderRecord, index)

If RTrim(OrderRecord.TableNumber) = cmbOpenOrders.Text And OrderRecord.OrderCleared = False Then 'Find the selected order

txtOrderID.Text = OrderRecord.OrderID 'Get employee record data

txtDate.Text = OrderRecord.OrderDate

For index2 = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord)

FileGet(EmployeeFileNumber, EmployeeRecord, index2)

If EmployeeRecord.EmployeeID = OrderRecord.EmployeeID Then

txtEmployeeName.Text = EmployeeRecord.EmployeeFirstName

End If

Next index2

txtTableNumber.Text = OrderRecord.TableNumber

rdbTableCleared.Checked = OrderRecord.OrderCleared

End If

Next index

CreateReports() 'Call procedure to populate list boxes

End Sub

Private Sub btnNewOrder\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnNewOrder.Click

txtOrderID.Text = "" 'Clear OrderID data box

txtDate.Text = Today.Date 'Enter todays date in the date box

txtEmployeeName.Text = GetEmployeeName() 'Call employee name function to get the name from the logged in ID

CreateReports() 'Populate list boxes

txtTableNumber.Text = "" 'Clear table number

rdbTableCleared.Checked = False 'Uncheck table cleared radio button

FileGet(IDFileNumber, IDRecord, OrderFileNumber)

txtOrderID.Text = IDRecord.ID + 1 'Create new unique ID

cmbOpenOrders.Text = ""

End Sub

Private Sub MenuComboLoad()

Dim MenuName As String

FileOpen(MenuFileNumber, MenuFilePath, OpenMode.Input) 'Open menu file

cmbMenu.Items.Clear()

Do While Not EOF(MenuFileNumber) 'Loop through every record

MenuName = LineInput(MenuFileNumber) 'Write menu name in combo box

cmbMenu.Items.Add(MenuName)

Loop

FileClose(MenuFileNumber) 'Close menu file

End Sub

Private Sub cmbMenu\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbMenu.SelectedIndexChanged

Dim index, NumberOfRecords As Integer

cmbMeal.Items.Clear() 'Clear combo box

NumberOfRecords = LOF(MealFileNumber) / Len(MealRecord)

For index = 1 To NumberOfRecords 'Loop through every record in the meal file

FileGet(MealFileNumber, MealRecord, index)

If RTrim(MealRecord.Menu) = cmbMenu.Text And MealRecord.InStock = True And MealRecord.Historic = False Then

cmbMeal.Items.Add(MealRecord.MealName) 'Add meals which aren't historic, are in stock and have the selected menu

End If

Next index

End Sub

Private Sub btnSaveRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAdd.Click

ValidateData() 'Call procedure to validate data

LoadOpenOrders() 'Call procedure to load open orders into aa combo box

End Sub

Private Sub ValidateData() 'Procedure to check the data entered is valid

Dim NumberOfOrderRecord As Integer = LOF(OrderFileNumber) / Len(OrderRecord)

Dim valid As Boolean = True

Dim Message As String = "" 'Strings to collect error messages

Dim Message2 As String = ""

Dim OrderSaved As Boolean = False 'Yes or no variable whether or not the order has already been created

Dim index As Integer

If txtOrderID.Text = "" Then 'If data entry field is empty add error message

MsgBox("Please Press New Meal to create a new Order")

Else

If txtTableNumber.Text = "" Then

Message = Message & "A Table Number" & vbNewLine

ElseIf IsNumeric(txtTableNumber.Text) = False Then 'Check table number is a number

Message2 = Message2 & "The table number you entered is not numeric"

Else

For Each item As Short In cmbOpenOrders.Items 'Check to see if the table number entered is a new table

If item = txtTableNumber.Text Then

For index = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'Loop through order records

FileGet(OrderFileNumber, OrderRecord, index)

If OrderRecord.OrderCleared = False And OrderRecord.TableNumber = txtTableNumber.Text Then 'If it's an open order and the same table number as an existing order

If OrderRecord.OrderID <> txtOrderID.Text Then 'If the existing order isn't the same as the new order

If MsgBox("This table is already in use, would you like to add this to the same table (Yes) or select a different table (No)", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then 'Check whether it should be added to a new table or part of this table.

txtOrderID.Text = OrderRecord.OrderID

txtDate.Text = OrderRecord.OrderDate

txtEmployeeName.Text = ""

Else

txtTableNumber.Text = (InputBox("Please enter a table number not in use"))

End If

End If

End If

Next index

End If

Next item

End If

If cmbMenu.Text = "" Then Message = Message & "A Menu" & vbNewLine 'If data entry box is empty write error message

If cmbMeal.Text = "" Then Message = Message & "A Meal" & vbNewLine

If txtQuantity.Text = "" Then

Message = Message & "A Quantity" & vbNewLine

ElseIf IsNumeric(txtQuantity.Text) = False Then 'Check quantity is a number

Message2 = Message2 & " The quantity entered is not a number"

End If

If Message = "" And Message2 = "" Then 'If Data is valid

For index = 1 To NumberOfOrderRecord 'Check to see if the order has been created

FileGet(OrderFileNumber, OrderRecord, index)

If OrderRecord.OrderID = txtOrderID.Text Then

OrderSaved = True

End If

Next

If OrderSaved = False Then 'If it's a new order call a procedure to save the order

SaveOrder()

End If

SaveOrderItem() 'Call procedure to save order item

txtQuantity.Text = "" 'Reset Quantity and meal text boxes

cmbMeal.Text = ""

CreateReports() 'Call procedure to populate the receipt and order reports

Else 'If error messages have no contents then don't save the message and display the error messages.

If Message2 = "" Then

MsgBox("Please Enter:" & vbNewLine & Message & vbNewLine & "This Meal has not been saved") 'Creating messages which make sense

ElseIf Message = "" Then

MsgBox("Some data was incorrectly entered:" & vbNewLine & Message2 & vbNewLine & "Please correct these " & vbNewLine & "This Meal has not been saved")

Else

MsgBox("Please Enter:" & vbNewLine & Message & vbNewLine & "Some data was incorrectly entered:" & vbNewLine & Message2 & vbNewLine & "Please correct this " & "This Meal has not been saved")

End If

End If

End If

End Sub

Private Sub SaveOrder()

OrderRecord.RecordNumber = LOF(OrderFileNumber) / Len(OrderRecord) + 1 'Move data from the input boxes into an order record

OrderRecord.OrderID = txtOrderID.Text

OrderRecord.OrderDate = txtDate.Text

OrderRecord.EmployeeID = CurrentEmployeeID

OrderRecord.TableNumber = txtTableNumber.Text

OrderRecord.OrderCleared = rdbTableCleared.Checked

FilePut(OrderFileNumber, OrderRecord, CInt(txtOrderID.Text)) 'Save record in the order file

IDRecord.FileNumber = OrderFileNumber 'Updte the Id file to contain the most recently used OrderID

IDRecord.ID = txtOrderID.Text

FilePut(IDFileNumber, IDRecord, OrderFileNumber)

End Sub

Private Sub SaveOrderItem()

Dim MealID, Index As Integer

Dim Price, Cost As Single

For Index = 1 To LOF(MealFileNumber) / Len(MealRecord) 'Retreiveing MealID, Price and Cost from the meal file

FileGet(MealFileNumber, MealRecord, Index)

If MealRecord.MealName = cmbMeal.Text Then

MealID = MealRecord.MealID

Price = MealRecord.PriceCurrent

Cost = MealRecord.CostCurrent

End If

Next

OrderItemRecord.OrderID = txtOrderID.Text 'Entering data from the meal lookup and input boxes to an orderitem record

OrderItemRecord.MealID = MealID

OrderItemRecord.Quantity = txtQuantity.Text

OrderItemRecord.Price = Price

OrderItemRecord.Cost = Cost

Dim NumberOfOrderItemRecords As Integer = LOF(OrderItemFileNumber) / Len(OrderItemRecord) 'Calculate number of records in the order item file

FilePut(OrderItemFileNumber, OrderItemRecord, NumberOfOrderItemRecords + 1) 'Place record in the order item file one place higher than the highest record

End Sub

Private Sub CreateReports()

Dim Index, Index2 As Integer

Dim MealName As String = ""

Dim Now As DateTime

Total = 0

Dim OrderItemTotal As Single

lstOrder.Items.Clear() 'Clear listboxes

lstReceipt.Items.Clear()

lstOrder.Items.Add("EASTNEY TAVERN")

lstOrder.Items.Add("")

lstOrder.Items.Add(txtDate.Text & " " & Now) 'Add date and current time

lstOrder.Items.Add("")

lstOrder.Items.Add(txtEmployeeName.Text)

lstOrder.Items.Add("")

lstReceipt.Items.Add("Eastney Tavern")

lstReceipt.Items.Add("")

lstReceipt.Items.Add(txtDate.Text & " " & Now)

lstReceipt.Items.Add("")

lstReceipt.Items.Add(txtEmployeeName.Text)

lstReceipt.Items.Add("")

If txtOrderID.Text <> "" Then

FileGet(OrderFileNumber, OrderRecord, (txtOrderID.Text)) 'Get current order Record

Dim NumberOfRecords As Integer = LOF(OrderItemFileNumber) / Len(OrderItemRecord)

For Index = 1 To NumberOfRecords 'For every order item record

FileGet(OrderItemFileNumber, OrderItemRecord, Index)

If OrderRecord.OrderID = OrderItemRecord.OrderID Then 'If the orderitemd record is part of the current order record

For Index2 = 1 To LOF(MealFileNumber) / Len(MealRecord) 'for every meal file

FileGet(MealFileNumber, MealRecord, Index2)

If OrderItemRecord.MealID = MealRecord.MealID Then 'Find the meal record with the same mealid as the current orderID

OrderItemTotal = OrderItemRecord.Price \* OrderItemRecord.Quantity 'Calculate total price of that meal (and quantity)

Total = Total + OrderItemTotal 'Add this to overall total

If MealRecord.MealName = "" Then

MealName = "Unknown meal"

Else

MealName = MealRecord.MealName

End If

End If

Next

lstOrder.Items.Add(OrderItemRecord.Quantity.ToString("D2") & " " & MealName) 'Enter this info into the order and receipt list box

lstReceipt.Items.Add(OrderItemRecord.Quantity.ToString("D2") & " " & MealName & " " & Format(OrderItemTotal, "Currency"))

End If

Next

lstReceipt.Items.Add("")

lstReceipt.Items.Add("Total " & Format(Total, "Currency")) 'Display total

End If

End Sub

Private Sub rdbTableCleared\_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles rdbTableCleared.CheckedChanged

If rdbTableCleared.Checked = True Then

If txtOrderID.Text <> "" Then 'As long as an order is selected

For index = 1 To (LOF(OrderFileNumber) / Len(OrderRecord)) 'For every order record

FileGet(OrderFileNumber, OrderRecord, index)

If OrderRecord.OrderID = CShort(txtOrderID.Text) Then 'If the order record is th current record

Dim RecordPosition As Integer = index

OrderRecord.OrderCleared = True 'Change the record so order cleaered is true

FilePut(OrderFileNumber, OrderRecord, RecordPosition) 'Resave the record

End If

Next

LoadOpenOrders() 'Call proced ure to load open orders

cmbOpenOrders.Text = "" 'Reset all input boxes to default

lstOrder.Items.Clear()

lstReceipt.Items.Clear()

txtDate.Text = ""

txtEmployeeName.Text = ""

txtOrderID.Text = ""

txtQuantity.Text = ""

txtTableNumber.Text = ""

End If

End If

End Sub

Private Sub lstReceipt\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles lstReceipt.SelectedIndexChanged

If lstReceipt.SelectedIndex < lstOrder.Items.Count Then 'Make selected lines in the receipt and order record the same

lstOrder.SelectedIndex = lstReceipt.SelectedIndex

End If

End Sub

Private Sub lstOrder\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles lstOrder.SelectedIndexChanged

lstReceipt.SelectedIndex = lstOrder.SelectedIndex 'Make selected lines in the Order and Receipt record the same

End Sub

Private Sub btnDeleteMeal\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDeleteMeal.Click

Dim OrderItem, MealName As String

Dim OrderID, MealID, Index, Index2, NoOfOrderItems As Integer

OrderItem = lstOrder.SelectedItem.ToString() 'Get OrderItem from the order listbox

If OrderItem <> "" Then

If IsNumeric(OrderItem.Substring(0, 3)) = True Then 'If the line has numeric data

MealName = OrderItem.Substring(3, 20) 'Extract meal name

OrderID = txtOrderID.Text

For Index = 1 To LOF(MealFileNumber) / Len(MealRecord) 'For every meal record

FileGet(MealFileNumber, MealRecord, Index)

If MealRecord.MealName = MealName Then

MealID = MealRecord.MealID 'Find meal record

End If

Next

End If

Else

MsgBox("select a meal by clicking on it in rder box or receipt box before trying to delete a meal")

End If

FileOpen(10, CurDir() & "/Temp.Dat", OpenMode.Random, , , Len(OrderItemRecord)) 'Open temporary file

For Index2 = 1 To LOF(OrderItemFileNumber) / Len(OrderItemRecord) 'For every order item record

FileGet(OrderItemFileNumber, OrderItemRecord, Index2)

If OrderItemRecord.OrderID <> txtOrderID.Text Or MealID <> OrderItemRecord.MealID Then 'If order item isn't part of the order you want deleted

FilePut(10, OrderItemRecord) 'Then save record to temporary file

End If

Next

FileClose(OrderItemFileNumber) 'Close both files

FileClose(10)

Kill(OrderItemFilePath) 'Delete order item file

FileCopy(CurDir() & "/Temp.Dat", OrderItemFilePath) 'Copy temp file to order items location

Kill(CurDir() & "/Temp.Dat") 'Delete temp file

FileOpen(OrderItemFileNumber, OrderItemFilePath, OpenMode.Random, , , Len(OrderItemRecord)) 'Reopen ammended orderitem file

CreateReports() 'Call procedures to create reports

End Sub

Private Sub btnLogOut\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnLogOut.Click

Me.Close()

CurrentEmployeeID = 0

frmLogin.Show()

End Sub

Private Sub btnDeleteOrder\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDeleteOrder.Click

Dim NoOfOrderItems As Integer = 0

FileOpen(10, CurDir() & "/Temp.Dat", OpenMode.Random, , , Len(OrderItemRecord)) 'Open temporary file

For Index2 = 1 To LOF(OrderItemFileNumber) / Len(OrderItemRecord) 'For every order item record

FileGet(OrderItemFileNumber, OrderItemRecord, Index2)

If OrderItemRecord.OrderID <> txtOrderID.Text Then 'If order item isn't part of the order you want deleted

NoOfOrderItems = NoOfOrderItems + 1 'Then save record to temporary file

FilePut(10, OrderItemRecord)

End If

FilePut(10, OrderItemRecord)

Next

FileClose(OrderItemFileNumber) 'Close both files

FileClose(10)

Kill(OrderItemFilePath) 'Delete order item file

FileCopy(CurDir() & "/Temp.Dat", OrderItemFilePath) 'Copy temp file to order items location

Kill(CurDir() & "/Temp.Dat") 'Delete temp file

FileOpen(OrderItemFileNumber, OrderItemFilePath, OpenMode.Random, , , Len(OrderItemRecord)) 'Reopen ammended orderitem file

DeleteOrder() 'Call procedure to delete order

End Sub

Sub DeleteOrder()

Dim Index, OrderRecordNumber As Integer

FileOpen(10, CurDir() & "/Temp.Dat", OpenMode.Random, , , Len(OrderItemRecord)) 'Create temporary file

For Index = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'For every order record

If OrderRecord.OrderID <> txtOrderID.Text Then 'If the order record from file is not the one wanting to be deleted

OrderRecordNumber = OrderRecordNumber + 1

FilePut(10, OrderRecord, OrderRecordNumber) 'Save recor din temp file

End If

Next Index

FileClose(OrderFileNumber) 'Close both files

FileClose(10)

Kill(OrderFilePath) 'Delete order file

FileCopy(CurDir() & "/Temp.Dat", OrderFilePath) 'Copy temp file to order file location

Kill(CurDir() & "/Temp.Dat") 'Delete temp file

FileOpen(OrderFileNumber, OrderFilePath, OpenMode.Random, , , Len(OrderRecord)) 'Reopen ammended order file

txtOrderID.Text = "" 'Set input boxes to default

txtDate.Text = Today.Date

txtEmployeeName.Text = ""

txtTableNumber.Text = ""

End Sub

'Printing

Private Sub btnPrintOrders\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPrintOrder.Click

If (PrintDialog1.ShowDialog() = Windows.Forms.DialogResult.OK) Then

PrintDialog1.Document.Print()

End If

End Sub

Private Sub PrintDocument1\_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage

Dim yMargin As Integer = e.MarginBounds.Y

Dim xMargin As Integer = e.MarginBounds.X

Dim currentpageItemProgress As Integer = 0

For Each item As String In lstOrder.Items

If (currentpageItemProgress >= ItemProgress) Then

e.Graphics.DrawString(item, lstOrder.Font, New SolidBrush(lstOrder.ForeColor), xMargin, yMargin)

yMargin += lstOrder.Font.Size + 10

ItemProgress += 1

If (yMargin >= e.MarginBounds.Y + e.MarginBounds.Height And ItemProgress <= lstOrder.Items.Count) Then

e.HasMorePages = True

currentpageItemProgress = 0

End If

End If

currentpageItemProgress += 1

Next

End Sub

Private Sub PrintDocument1\_EndPrint(ByVal sender As Object, ByVal e As System.Drawing.Printing.PrintEventArgs) Handles PrintDocument1.EndPrint

ItemProgress = 0

End Sub

Private Sub btnPrintReceipt\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPrintReceipt.Click

If (PrintDialog2.ShowDialog() = Windows.Forms.DialogResult.OK) Then

PrintDialog2.Document.Print()

End If

End Sub

Private Sub PrintDocument2\_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument2.PrintPage

Dim yMargin As Integer = e.MarginBounds.Y

Dim xMargin As Integer = e.MarginBounds.X

Dim currentpageItemProgress As Integer = 0

For Each item As String In lstReceipt.Items

If (currentpageItemProgress >= ItemProgress) Then

e.Graphics.DrawString(item, lstReceipt.Font, New SolidBrush(lstReceipt.ForeColor), xMargin, yMargin)

yMargin += lstReceipt.Font.Size + 10

ItemProgress += 1

If (yMargin >= e.MarginBounds.Y + e.MarginBounds.Height And ItemProgress <= lstReceipt.Items.Count) Then

e.HasMorePages = True

currentpageItemProgress = 0

End If

End If

currentpageItemProgress += 1

Next

End Sub

Private Sub PrintDocument2\_EndPrint(ByVal sender As Object, ByVal e As System.Drawing.Printing.PrintEventArgs) Handles PrintDocument2.EndPrint

ItemProgress = 0

End Sub

Private Sub btnPay\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPay.Click

Dim AmountPaid As String

If MsgBox("Pay with cash?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

AmountPaid = InputBox("Money Paid")

If IsNumeric(AmountPaid) = True Then

If CInt(AmountPaid) - Total > 0 Then

If MsgBox("Give £" & CInt(AmountPaid) - Total & " Change" & vbNewLine & "Has this table been cleared?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

rdbTableCleared.Checked = True

End If

Else

MsgBox("The money given isn't enough to cover the bill" & vbNewLine & "Collect £" & Total - CInt(AmountPaid))

End If

Else

MsgBox("Sorry the amount you entered was not a number")

End If

ElseIf MsgBox("Has this table been cleared?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

rdbTableCleared.Checked = True

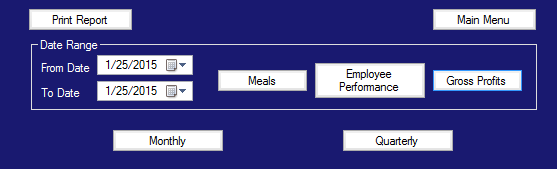
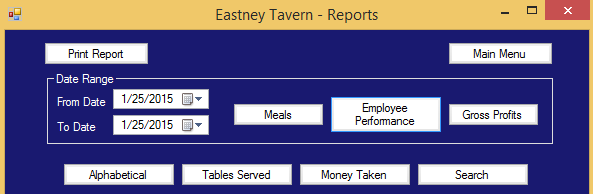
End If

End Sub

End Class

## Reports Form





Public Class frmReports

Dim OrdersExist As Boolean = False 'Global variable to declare if the order exists

Public Structure MealStructure 'Self assigned variable for data stored in array

<VBFixedString(20)> Dim MealName As String

Dim NumberOfMeals As Short

Dim MoneyTaken As Single

Dim Profit As Single

End Structure

Public Structure EmployeeStructure

<VBFixedString(31)> Dim Name As String

Dim NumberOfOrders As Short

Dim NumberOfMeals As Short

Dim MoneyTaken As Single

End Structure

Dim Meals((LOF(MealFileNumber) / Len(MealRecord)) + 1) As MealStructure 'Create arrays with custom file type

Dim Employees(LOF(EmployeeFileNumber) / Len(EmployeeRecord) + 1) As EmployeeStructure

Dim FormatEmployeeReport As String = "{0,-32}{1,-10}{2,-10}{3,-7}" 'Custom Formats

Dim FormatMealsReport As String = "{0,-21}{1,-9}{2,-10}{3,-8}"

Private Sub frmReports\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

If LOF(OrderFileNumber) / Len(OrderRecord) <> 0 Then 'If orders have been taken add dates to date box (form the first order taken to the last order taken)

FileGet(OrderFileNumber, OrderRecord, 1)

datFromDate.MinDate = OrderRecord.OrderDate

datToDate.MinDate = OrderRecord.OrderDate

FileGet(OrderFileNumber, OrderRecord, LOF(OrderFileNumber) / Len(OrderRecord))

datToDate.MaxDate = OrderRecord.OrderDate

datFromDate.MaxDate = OrderRecord.OrderDate

datToDate.Text = OrderRecord.OrderDate

Else 'If there's no orders than disable the date entry box

datFromDate.Enabled = False

datToDate.Enabled = False

End If

End Sub

Private Sub PopulateEmployeeArray() 'Procedure to take data from order file and get info about each employee

Dim index1, index2, index3, NumberOfOrders, NumberOfMeals As Integer

Dim MoneyTaken As Double

Dim DeletedEmployeeID As Short

FileGet(IDFileNumber, IDRecord, EmployeeFileNumber)

If IDRecord.ID >= LOF(EmployeeFileNumber) / Len(EmployeeRecord) Then 'Checks to see if records have been deleted

Dim UnKnownNumberOfOrders As Single = 0

Dim UnKnownNumberOfMeals As Integer = 0

Dim UnKnownMoneyTaken As Single = 0

FileOpen(DeletedEmployeeIDFileNumber, DeletedEmployeeIDFilePath, OpenMode.Input) 'Open file of deleted employee IDs

Do While Not EOF(DeletedEmployeeIDFileNumber) 'For every deleted emplyee ID

DeletedEmployeeID = CShort(LineInput(DeletedEmployeeIDFileNumber))

For index2 = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'Search every order record to find the deleted employee ID

FileGet(OrderFileNumber, OrderRecord, index2)

If OrderRecord.OrderDate >= CDate(datFromDate.Text) And OrderRecord.OrderDate <= CDate(datToDate.Text) And DeletedEmployeeID = OrderRecord.EmployeeID Then 'Check date of order is within inputted parammeters

UnKnownNumberOfOrders = UnKnownNumberOfOrders + 1

For index3 = 1 To LOF(OrderItemFileNumber) / Len(OrderItemRecord)

FileGet(OrderItemFileNumber, OrderItemRecord, index3)

If OrderRecord.OrderID = OrderItemRecord.OrderID Then

UnKnownNumberOfMeals = UnKnownNumberOfMeals + OrderItemRecord. 'Increase total unknown number of meals and unknown money taken

UnKnownMoneyTaken = UnKnownMoneyTaken + (OrderItemRecord.Quantity \* OrderItemRecord.Price)

End If

Next index3

End If

Next

Loop

FileClose(DeletedEmployeeIDFileNumber) 'Close deletd employee ID file

Employees(LOF(EmployeeFileNumber) / Len(EmployeeRecord) + 1).Name = "Unknown Employee" 'Place information created into the last position in the array

Employees(LOF(EmployeeFileNumber) / Len(EmployeeRecord) + 1).NumberOfMeals = UnKnownNumberOfMeals

Employees(LOF(EmployeeFileNumber) / Len(EmployeeRecord) + 1).MoneyTaken = UnKnownMoneyTaken

Employees(LOF(EmployeeFileNumber) / Len(EmployeeRecord) + 1).NumberOfOrders = UnKnownNumberOfOrders

End If

For index1 = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord) 'For every employee record

NumberOfOrders = 0

NumberOfMeals = 0

MoneyTaken = 0

FileGet(EmployeeFileNumber, EmployeeRecord, index1)

For index2 = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'Loop through every order

FileGet(OrderFileNumber, OrderRecord, index2)

If OrderRecord.EmployeeID = EmployeeRecord.EmployeeID And OrderRecord.OrderDate >= CDate(datFromDate.Text) And OrderRecord.OrderDate <= CDate(datToDate.Text) Then 'Check order is within date parameteters

NumberOfOrders = NumberOfOrders + 1

For index3 = 1 To LOF(OrderItemFileNumber) / Len(OrderItemRecord) 'Loop through every order item

FileGet(OrderItemFileNumber, OrderItemRecord, index3)

If OrderRecord.OrderID = OrderItemRecord.OrderID Then

NumberOfMeals = NumberOfMeals + OrderItemRecord.Quantity 'Total up the number of meals

MoneyTaken = MoneyTaken + OrderItemRecord.Quantity \* OrderItemRecord.Price 'Total up the money taken

End If

Next index3

End If

Next index2

Employees(index1).Name = RTrim(EmployeeRecord.EmployeeFirstName) & " " & RTrim(EmployeeRecord.EmployeeLastName) 'Add info to the employee array

Employees(index1).NumberOfOrders = NumberOfOrders

Employees(index1).NumberOfMeals = NumberOfMeals

Employees(index1).MoneyTaken = MoneyTaken

Next index1

End Sub

Private Sub PopluateMealArray() 'Procedure to take data from order file and get info about each meal

ReDim Meals((LOF(MealFileNumber) / Len(MealRecord)) + 1)

Dim index1, index2, index3, NumberOfMeals As Integer

Dim Profit, MoneyTaken As Double

For index1 = 1 To LOF(MealFileNumber) / Len(MealRecord) 'For every meal record

NumberOfMeals = 0

MoneyTaken = 0

Profit = 0

FileGet(MealFileNumber, MealRecord, index1)

For index2 = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'Loop through every order record

FileGet(OrderFileNumber, OrderRecord, index2)

If OrderRecord.OrderDate >= CDate(datFromDate.Text) And OrderRecord.OrderDate <= CDate(datToDate.Text) Then 'Check order is within the date parameters

For index3 = 1 To LOF(OrderItemFileNumber) / Len(OrderItemRecord) 'Check every order item record

FileGet(OrderItemFileNumber, OrderItemRecord, index3)

If OrderRecord.OrderID = OrderItemRecord.OrderID And MealRecord.MealID = OrderItemRecord.MealID Then 'If it's the right order and meal

NumberOfMeals = NumberOfMeals + OrderItemRecord.Quantity 'total number of meals, profit and money taken with the same Meal ID

Profit = Profit + (OrderItemRecord.Price - OrderItemRecord.Cost) \* OrderItemRecord.Quantity

MoneyTaken = MoneyTaken + (OrderItemRecord.Quantity \* OrderItemRecord.Price)

End If

Next index3

End If

Next

If MealRecord.Historic = True And NumberOfMeals = 0 Then 'Old meals with no orders placed

Meals(index1).MealName = ""

Else

Meals(index1).MealName = MealRecord.MealName

End If

Meals(index1).NumberOfMeals = NumberOfMeals 'Place data into array

Meals(index1).MoneyTaken = MoneyTaken

Meals(index1).Profit = Profit

Next index1

End Sub

Private Sub btnMainMenu\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMainMenu.Click

Me.Close()

frmMenu.Show()

End Sub

Private Sub btnEmployeePerformance\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEmployeePerformance.Click

lstReports.Items.Clear()

If LOF(OrderFileNumber) / Len(OrderRecord) = 0 Then 'If no orders have been taken

MsgBox("Unfortunately you can't create reports with no orders taken.") 'Display error message

Else

PopulateEmployeeArray() 'Call procedure to populat employee array

btnMealAlphabetical.Visible = False 'Hide all buttons except those relevant to employss info

btnMealsByPopularity.Visible = False

btnMealSearch.Visible = False

btnByProfit.Visible = False

btnEmployeeSearch.Visible = True

btnEmployeesAlphabetical.Visible = True

btnMoneyTaken.Visible = True

btnTablesServed.Visible = True

btnQarterly.Visible = False

btnMonthly.Visible = False

End If

End Sub

Private Sub DisplayMealList()

Dim TotalMeals, TotalMoney, TotalProfit As Single

lstReports.Items.Add("Meal Name Number Money Profit ") 'Display column headings

lstReports.Items.Add(" of Meals Taken")

lstReports.Items.Add("================================================")

For index1 = 1 To (LOF(MealFileNumber) / Len(MealRecord)) + 1 'Loop thrugh employee array

If Meals(index1).MealName <> "" Then

TotalMeals = TotalMeals + Meals(index1).NumberOfMeals 'Calculate totals

TotalMoney = TotalMoney + Meals(index1).MoneyTaken

TotalProfit = TotalProfit + Meals(index1).Profit

lstReports.Items.Add(String.Format(FormatMealsReport, (Meals(index1).MealName), (Meals(index1).NumberOfMeals), (Format(Meals(index1).MoneyTaken, "Currency")), (Format(Meals(index1).Profit, "Currency")))) 'display info in custom format

End If

Next

lstReports.Items.Add("")

lstReports.Items.Add(String.Format("{0,-21}{1,-9}{2,-10}{3,-10}", "Total", TotalMeals, (Format(TotalMoney, "Currency")), (Format(TotalProfit, "Currency")))) 'Display totals

End Sub

Private Sub btnMealAlphabetical\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMealAlphabetical.Click 'Procedure reorders the meal array (A-->Z)

Dim Temp As MealStructure

lstReports.Items.Clear()

For index2 = 1 To (LOF(MealFileNumber) / Len(MealRecord)) + 1 'Loop through meal array twice

For index = 1 To (LOF(MealFileNumber) / Len(MealRecord))

If Meals(index).MealName > Meals(index + 1).MealName Then 'If the 1st meal in the array is later in the alphabet than the second meal

Temp = Meals(index) 'Place first meal in a temporary variable

Meals(index) = Meals(index + 1) 'replace frist meal with second

Meals(index + 1) = Temp 'Place temporary (1st meal) in second meal

End If

Next

Next index2

DisplayMealList() 'Call procedure to display the array

End Sub

Private Sub btnMealsByPopularity\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMealsByPopularity.Click 'Procedure reorders the meal array (By number of meals sold)

Dim Temp As MealStructure

lstReports.Items.Clear()

For index2 = 1 To (LOF(MealFileNumber) / Len(MealRecord)) + 1 'Loop through meal array twice

For index = 1 To (LOF(MealFileNumber) / Len(MealRecord))

If Meals(index + 1).MealName <> "" And Meals(index).NumberOfMeals < Meals(index + 1).NumberOfMeals Then 'If the 1st meal in the array is less than the second meal

Temp = Meals(index) 'Place first meal in a temporary variable

Meals(index) = Meals(index + 1) 'replace frist meal with second

Meals(index + 1) = Temp 'Place temporary (1st meal) in second meal

End If

Next

Next index2

DisplayMealList() 'Call procedure to display the array

End Sub

Private Sub btnByProfit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnByProfit.Click 'Procedure reorders the meal array (By Profit)

Dim Temp As MealStructure

Dim NoMoreSwaps As Boolean

lstReports.Items.Clear()

NoMoreSwaps = True

For index = 1 To (LOF(MealFileNumber) / Len(MealRecord)) + 1

For index2 = 1 To (LOF(MealFileNumber) / Len(MealRecord))

If Meals(index2).Profit < Meals(index2 + 1).Profit Then

Temp = Meals(index2) 'Place first meal in a temporary variable

Meals(index2) = Meals(index2 + 1) 'Replace frist meal with second

Meals(index2 + 1) = Temp 'Place temporary (1st meal) in second meal

End If

Next index2

Next index

DisplayMealList() 'Call procedure to display the array

End Sub

Private Sub btnEmployeesAlphabetical\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEmployeesAlphabetical.Click 'Procedure reorders the employee array (Alphabetically)

Dim Temp As EmployeeStructure

lstReports.Items.Clear()

For index2 = 1 To (LOF(EmployeeFileNumber) / Len(EmployeeRecord))

For index = 1 To (LOF(EmployeeFileNumber) / Len(EmployeeRecord)) - 1

If Employees(index).Name > Employees(index + 1).Name Then

Temp = Employees(index) 'Place first meal in a temporary variable

Employees(index) = Employees(index + 1) 'Replace frist meal with second

Employees(index + 1) = Temp 'Place temporary (1st meal) in second meal

End If

Next

Next index2

DisplayEmployeeList() 'Call procedure to display the array

End Sub

Private Sub btnTablesServed\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnTablesServed.Click 'Procedure reorders the Employee array (By Tabbles served)

Dim Temp As EmployeeStructure

lstReports.Items.Clear()

For index2 = 1 To (LOF(EmployeeFileNumber) / Len(EmployeeRecord))

For index = 1 To (LOF(EmployeeFileNumber) / Len(EmployeeRecord)) - 1

If Employees(index).NumberOfOrders < Employees(index + 1).NumberOfOrders Then

Temp = Employees(index) 'Place first meal in a temporary variable

Employees(index) = Employees(index + 1) 'Replace frist meal with second

Employees(index + 1) = Temp 'Place temporary (1st meal) in second meal

End If

Next

Next index2

DisplayEmployeeList() 'Call procedure to display the array

End Sub

Private Sub btnMoneyTaken\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMoneyTaken.Click 'Procedure reorders the emplyee array (By Money Taken)

Dim Temp As EmployeeStructure

lstReports.Items.Clear()

For index2 = 1 To (LOF(EmployeeFileNumber) / Len(EmployeeRecord))

For index = 1 To (LOF(EmployeeFileNumber) / Len(EmployeeRecord)) - 1

If Employees(index).MoneyTaken < Employees(index + 1).MoneyTaken Then

Temp = Employees(index) 'Place first meal in a temporary variable

Employees(index) = Employees(index + 1) 'Replace frist meal with second

Employees(index + 1) = Temp 'Place temporary (1st meal) in second meal

End If

Next

Next index2

DisplayEmployeeList() 'Call procedure to display the array

End Sub

Private Sub DisplayEmployeeList()

Dim TotalOrders, TotalMeals, TotalMoney As Single

lstReports.Items.Add("Employee Name Number of Number of Money") 'Display column headings

lstReports.Items.Add(" Orders Meals Taken")

lstReports.Items.Add("=============================================================")

For index1 = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord) + 1 'For every employee in the employees array

If Employees(index1).Name <> "" Then

lstReports.Items.Add(String.Format(FormatEmployeeReport, (Employees(index1).Name), (Employees(index1).NumberOfOrders), (Employees(index1).NumberOfMeals), (Format(Employees(index1).MoneyTaken, "Currency")))) 'Write the information to the listbox with custom format

TotalMeals = TotalMeals + Employees(index1).NumberOfMeals 'Keep running adding totals

TotalMoney = TotalMoney + Employees(index1).MoneyTaken

TotalOrders = TotalOrders + Employees(index1).NumberOfOrders

End If

Next

lstReports.Items.Add("")

lstReports.Items.Add(String.Format("{0,-32}{1,-10}{2,-10}{3,-10}", "Total", TotalOrders, TotalMeals, (Format(TotalMoney, "Currency")))) 'Display totals

End Sub

Private Sub btnMeals\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMeals.Click

lstReports.Items.Clear()

If LOF(OrderFileNumber) / Len(OrderRecord) = 0 Then 'If Orders haven't been taken

MsgBox("Unfortunately you can't create reports with no orders taken.") 'Display error messag

Else

PopluateMealArray() 'Call procedure to populat meal array

btnByProfit.Visible = True 'Hide all buttons unrelated to Meals

btnMealAlphabetical.Visible = True

btnMealsByPopularity.Visible = True

btnMealSearch.Visible = True

btnEmployeeSearch.Visible = False

btnEmployeesAlphabetical.Visible = False

btnMoneyTaken.Visible = False

btnTablesServed.Visible = False

btnQarterly.Visible = False

btnMonthly.Visible = False

End If

End Sub

Private Sub btnMealSearch\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMealSearch.Click

Dim SearchName As String

Dim Found As Boolean = False

SearchName = InputBox("Enter a Meal name to search") 'Get SearchName variable

lstReports.Items.Clear()

If SearchName <> "" Then

lstReports.Items.Add("Meal Name Number Money Profit") 'Display collumn headings

lstReports.Items.Add(" of Meals Taken")

lstReports.Items.Add("==============================================")

For index1 = 1 To LOF(MealFileNumber) / Len(MealRecord) 'For every Meal record compare the name to the searched name

If SearchName.ToLower = RTrim(Meals(index1).MealName).ToLower Then 'With all letters lowercase

lstReports.Items.Add(String.Format(FormatMealsReport, (Meals(index1).MealName), (Meals(index1).NumberOfMeals), (Format(Meals(index1).MoneyTaken, "Currency")), (Format(Meals(index1).Profit, "Currency")))) 'If found, display meals info

Found = True

Else

Dim FullMealName As String() = RTrim((Meals(index1).MealName)).Split(New Char() {" "c}) 'Compare each word in the meal name with the searched word

Dim word As String

For Each word In FullMealName

If SearchName.ToLower = word.ToLower Then

lstReports.Items.Add(String.Format(FormatMealsReport, (Meals(index1).MealName), (Meals(index1).NumberOfMeals), (Format(Meals(index1).MoneyTaken, "Currency")), (Format(Meals(index1).Profit, "Currency")))) 'If found, display meals info

Found = True

End If

Next

End If

Next index1

End If

If Found = False Then 'If nothing found

lstReports.Items.Add("")

lstReports.Items.Add("")

lstReports.Items.Add("Sorry no Meals were found with this name.") 'Display error message

lstReports.Items.Add("Please check you spelt the Meal name correctly")

lstReports.Items.Add("and that this Meal is in the system.")

End If

End Sub

Private Sub btnEmployeeSearch\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEmployeeSearch.Click

Dim SearchName As String

Dim Found As Boolean = False

lstReports.Items.Clear()

SearchName = InputBox("Enter an Employee name to search") 'Get SearchName variable

If SearchName <> "" Then

lstReports.Items.Add("Employee Name Number of Number of Money") 'Display collumn headings

lstReports.Items.Add(" Orders Meals Taken")

lstReports.Items.Add("=============================================================")

For index1 = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord) 'For every Meal record compare the name to the searched name

If SearchName.ToLower = (Employees(index1).Name).ToLower Then 'With all letters lowercase

lstReports.Items.Add(String.Format(FormatEmployeeReport, (Employees(index1).Name), (Employees(index1).NumberOfOrders), (Employees(index1).NumberOfMeals), (Format(Employees(index1).MoneyTaken, "Currency")))) 'If found, display employees info

Found = True

Else

Dim FullNames As String() = (Employees(index1).Name).Split(New Char() {" "c}) 'Compare each word in the Employee name with the searched word

Dim Name As String

For Each Name In FullNames

If SearchName.ToLower = Name.ToLower Then

lstReports.Items.Add(String.Format(FormatEmployeeReport, (Employees(index1).Name), (Employees(index1).NumberOfOrders), (Employees(index1).NumberOfMeals), (Format(Employees(index1).MoneyTaken, "Currency")))) 'If found, display employees info

Found = True

End If

Next

End If

Next index1

If Found = False Then 'If nothing found

lstReports.Items.Add("")

lstReports.Items.Add("")

lstReports.Items.Add("Sorry no Employees were found with this name.") 'Display error message

lstReports.Items.Add("Please check you spelt the name correctly")

lstReports.Items.Add("and that this Employee is in the system.")

End If

End If

End Sub

Private Sub datFromDate\_ValueChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles datFromDate.ValueChanged

btnEmployeeSearch.Visible = False 'Hide all sub serch buttons

btnEmployeesAlphabetical.Visible = False

btnMoneyTaken.Visible = False

btnTablesServed.Visible = False

btnByProfit.Visible = False

btnMealAlphabetical.Visible = False

btnMealsByPopularity.Visible = False

btnMealSearch.Visible = False

btnQarterly.Visible = False

btnMonthly.Visible = False

lstReports.Items.Clear() 'Clear report

End Sub

Private Sub datToDate\_ValueChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles datToDate.ValueChanged

btnEmployeeSearch.Visible = False 'Hide all sub serch buttons

btnEmployeesAlphabetical.Visible = False

btnMoneyTaken.Visible = False

btnTablesServed.Visible = False

btnByProfit.Visible = False

btnMealAlphabetical.Visible = False

btnMealsByPopularity.Visible = False

btnMealSearch.Visible = False

btnQarterly.Visible = False

btnMonthly.Visible = False

lstReports.Items.Clear() 'Clear report

End Sub

Private Sub btnPrint\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPrint.Click 'Printing

If (PrintDialog1.ShowDialog() = Windows.Forms.DialogResult.OK) Then

PrintDialog1.Document.Print()

End If

End Sub

Private Sub PrintDocument1\_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage

Dim yMargin As Integer = e.MarginBounds.Y

Dim xMargin As Integer = e.MarginBounds.X

Dim currentpageItemProgress As Integer = 0

For Each item As String In lstReports.Items

If (currentpageItemProgress >= ItemProgress) Then

e.Graphics.DrawString(item, lstReports.Font, New SolidBrush(lstReports.ForeColor), xMargin, yMargin)

yMargin += lstReports.Font.Size + 10

ItemProgress += 1

If (yMargin >= e.MarginBounds.Y + e.MarginBounds.Height And ItemProgress <= lstReports.Items.Count) Then

e.HasMorePages = True

currentpageItemProgress = 0

End If

End If

currentpageItemProgress += 1

Next

End Sub

Private Sub PrintDocument1\_EndPrint(ByVal sender As Object, ByVal e As System.Drawing.Printing.PrintEventArgs) Handles PrintDocument1.EndPrint

ItemProgress = 0

End Sub

Private Sub btnGrossProfit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGrossProfit.Click

lstReports.Items.Clear() 'Cear report

btnQarterly.Visible = True 'Hide all buttons which don;t elate to Gross proffit

btnMonthly.Visible = True

btnByProfit.Visible = False

btnMealAlphabetical.Visible = False

btnMealsByPopularity.Visible = False

btnMealSearch.Visible = False

btnEmployeeSearch.Visible = False

btnEmployeesAlphabetical.Visible = False

btnMoneyTaken.Visible = False

btnTablesServed.Visible = False

PopulateGrossProfitArray() 'Call procedure to populate gross proffit array

End Sub

Public Structure GrossProfit 'Define custom structure

Dim Month As Short

Dim Year As Short

Dim MoneyTaken As Single

Dim Profit As Single

End Structure

Dim Profits() As GrossProfit 'Create empty array

Public NumberOfMonths As Integer

Private Sub PopulateGrossProfitArray()

Dim OrderDate As Date

Dim index1, index3, CurrentYear, RecordMonth, RecordYear, ArrayPosition As Integer

Dim CurrentMonth As Integer = 0

Dim Profit, MoneyTaken As Double

For index1 = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'for every record in the order file

FileGet(OrderFileNumber, OrderRecord, index1)

OrderDate = CDate(OrderRecord.OrderDate)

If OrderRecord.OrderDate >= CDate(datFromDate.Text) And OrderRecord.OrderDate <= CDate(datToDate.Text) Then 'If order is withing the given dates

RecordMonth = Month(OrderDate)

RecordYear = Year(OrderDate)

If CurrentMonth = 0 Then 'For the first month

CurrentMonth = RecordMonth

CurrentYear = RecordYear

ArrayPosition = 1

End If

If RecordMonth <> CurrentMonth Or RecordYear <> CurrentYear Then 'If the record month is not the crrent month

ReDim Preserve Profits(ArrayPosition) 'Resize the array keeping the old data

Profits(ArrayPosition).Month = CurrentMonth 'Place data in array

Profits(ArrayPosition).Year = CurrentYear

Profits(ArrayPosition).MoneyTaken = MoneyTaken

Profits(ArrayPosition).Profit = Profit

MoneyTaken = 0 'Reset monthly variables

Profit = 0

CurrentMonth = RecordMonth 'update curren month/year to that of the record

CurrentYear = RecordYear

ArrayPosition = ArrayPosition + 1 'Increment array position by one

End If

For index3 = 1 To LOF(OrderItemFileNumber) / Len(OrderItemRecord) 'Loop through order item file

FileGet(OrderItemFileNumber, OrderItemRecord, index3)

If OrderRecord.OrderID = OrderItemRecord.OrderID Then 'Find the Id in the current onth

Profit = Profit + (OrderItemRecord.Price - OrderItemRecord.Cost) \* OrderItemRecord.Quantity 'update profit and moneytaken varuables

MoneyTaken = MoneyTaken + (OrderItemRecord.Quantity \* OrderItemRecord.Price)

End If

Next index3

End If

If index1 = LOF(OrderFileNumber) / Len(OrderRecord) Then 'for the last record

ReDim Preserve Profits(ArrayPosition)

Profits(ArrayPosition).Month = CurrentMonth 'Save data into array

Profits(ArrayPosition).Year = CurrentYear

Profits(ArrayPosition).MoneyTaken = MoneyTaken

Profits(ArrayPosition).Profit = Profit

MoneyTaken = 0

Profit = 0

CurrentMonth = RecordMonth

CurrentYear = RecordYear

ArrayPosition = ArrayPosition + 1

End If

Next index1

NumberOfMonths = ArrayPosition - 1 'Number of months corrected to hoow many are saved in the record.

End Sub

Public FormatMonthlyProfit As String = "{0,-10}{1,-5}{2,-10}{3,-10}"

Private Sub btnMonthly\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMonthly.Click

Dim TotalMoney, TotalProfits As Single

Dim MonthName As String = ""

lstReports.Items.Clear()

lstReports.Items.Add(String.Format(FormatMonthlyProfit, "Month", "Year", "Money", "Profits")) 'Display column tittles

lstReports.Items.Add(String.Format("{0,20}", "Taken"))

lstReports.Items.Add("==================================")

For index1 = 1 To NumberOfMonths 'Case slector to get mont nam from month number

Select Case Profits(index1).Month

Case 1

MonthName = "January"

Case 2

MonthName = "February"

Case 3

MonthName = "March"

Case 4

MonthName = "April"

Case 5

MonthName = "May"

Case 6

MonthName = "June"

Case 7

MonthName = "July"

Case 8

MonthName = "August"

Case 9

MonthName = "September"

Case 10

MonthName = "October"

Case 11

MonthName = "November"

Case 12

MonthName = "December"

End Select

TotalMoney = TotalMoney + Profits(index1).MoneyTaken 'Update totals

TotalProfits = TotalProfits + Profits(index1).Profit

lstReports.Items.Add(String.Format(FormatMonthlyProfit, MonthName, Profits(index1).Year, (Format(Profits(index1).MoneyTaken, "Currency")), (Format(Profits(index1).Profit, "Currency")))) 'display information in custom format

Next

lstReports.Items.Add("")

lstReports.Items.Add(String.Format("{0,-15}{1,-10}{2,-10}", "Total", (Format(TotalMoney, "Currency")), Format(TotalProfits, "Currency"))) 'Display totals

End Sub

Private Sub btnQarterly\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnQarterly.Click

Dim CurrentQuarter As String = ""

Dim RecordQuarter As String = ""

Dim Profit, MoneyTaken, TotalProfit, TotalMoney As Single

lstReports.Items.Clear()

lstReports.Items.Add(String.Format(FormatMonthlyProfit, "Quarter", "Year", "Money", "Profits")) 'Display tittle headings

lstReports.Items.Add(String.Format("{0,20}", "Taken"))

lstReports.Items.Add("==================================")

For index1 = 1 To NumberOfMonths

TotalProfit = TotalProfit + Profits(index1).Profit

TotalMoney = TotalMoney + Profits(index1).MoneyTaken

Select Case Profits(index1).Month 'Case selctor used to group every three months as a quarter

Case 1, 2, 3

RecordQuarter = "First"

Case 4, 5, 6

RecordQuarter = "Second"

Case 7, 8, 9

RecordQuarter = "Third"

Case 10, 11, 12

RecordQuarter = "Fourth"

End Select

If CurrentQuarter = "" Then 'First month in array

CurrentQuarter = RecordQuarter

End If

If CurrentQuarter = RecordQuarter Then 'Middle month in quarter

Profit = Profits(index1).Profit + Profit

MoneyTaken = Profits(index1).MoneyTaken + MoneyTaken

Else 'Last month in quarter

lstReports.Items.Add(String.Format(FormatMonthlyProfit, CurrentQuarter, Profits(index1).Year, (Format(MoneyTaken, "Currency")), (Format(Profit, "Currency"))))

CurrentQuarter = RecordQuarter

Profit = 0

MoneyTaken = 0

Profit = Profits(index1).Profit + Profit

MoneyTaken = Profits(index1).MoneyTaken + MoneyTaken

End If

If index1 = NumberOfMonths Then 'Display last month even if not the last month in quarter

lstReports.Items.Add(String.Format(FormatMonthlyProfit, CurrentQuarter, Profits(index1).Year, (Format(MoneyTaken, "Currency")), (Format(Profit, "Currency"))))

End If

Next

lstReports.Items.Add("")

lstReports.Items.Add(String.Format("{0,-15}{1,-10}{2,-10}", "Total", (Format(TotalMoney, "Currency")), Format(TotalProfit, "Currency"))) 'Display totals

End Sub

End Class

## Employees Form

Public Class frmEmployees

Private Sub frmEmployees\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

CreateReport()

If FirstUse = True Then

MsgBox("Please create an administrative employee account (with a password")

btnMainMenu.Visible = False

btnDeleteRecord.Visible = False

btnLastRecord.Visible = False

btnNextRecord.Visible = False

grpSearch.Visible = False

btnPrint.Visible = False

btnClose.Visible = True

End If

End Sub

Private Sub btnMainMenu\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMainMenu.Click

Me.Close()

frmMenu.Show()

End Sub

Private Sub btnNewEmployee\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnNewEmployee.Click

FileGet(IDFileNumber, IDRecord, EmployeeFileNumber)

txtEmployeeID.Text = IDRecord.ID + 1 'Get the next unused ID

txtFirstName.Text = "" 'Clear Data entry boxes

txtLastName.Text = ""

txtJobTittle.Text = ""

txtPayRate.Text = ""

txtPhoneNumber.Text = ""

txtEmail.Text = ""

txtAddress.Text = ""

txtECName.Text = ""

txtECPhoneNumber.Text = ""

txtPassword.Text = ""

txtFirstName.Focus()

End Sub

Private Sub btnSaveRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSaveRecord.Click

ValidateData() 'Calls the procedure to validate data

CreateReport() 'Calls the procedure to populate the listbox

End Sub

Private Sub ValidateData()

If txtEmployeeID.Text = "" Then 'If a Meal ID has't been created display an error.

MsgBox("You must have an Employee ID before you can save a new employee." & vbNewLine & "To do this you must click the 'New Employee' button" & vbNewLine & "This employee has not been saved.")

Else

Dim Message As String = "" 'Messages used to create longer strings with instructions on how to make the system save data.

Dim Message2 As String = ""

Dim Message3 As String = ""

If txtFirstName.Text = "" Then Message = "A First Name" & vbNewLine 'Validations to check text boxes contain data.

If txtLastName.Text = "" Then Message = "A Last Name" & vbNewLine

If txtJobTittle.Text = "" Then Message = "A Job Tittle " & vbNewLine

If txtPayRate.Text = "" Then

Message = "A Pay Rate" & vbNewLine

ElseIf IsNumeric(txtPayRate.Text) = False Then 'Validating the Numbers entered are actually numbers

Message2 = Message2 & "The Pay rate isn't a number" & vbNewLine

End If

If txtPhoneNumber.Text = "" Then

Message = Message & "A Phone Number" & vbNewLine

ElseIf IsNumeric(txtPhoneNumber.Text) = False Then

Message2 = Message2 & "The Phone Number isn't a number" & vbNewLine

ElseIf Len(txtPhoneNumber.Text) > 13 Then

Message3 = "The phone number you entered is not the correct length" & vbNewLine & "A phone number must be 13 numbers with no spaces"

End If

If txtEmail.Text = "" Then 'Checking it has a standard email format (Contains @)

Message = Message & "An Email" & vbNewLine

Else

Dim AtSign As Boolean

Dim Character As Char

Dim Email As String = txtEmail.Text

For Each Character In Email

If Character = "@" Then

AtSign = True

End If

Next

If AtSign = False Then

Message2 = Message2 & "The email doesn't contain an @ sign" & vbNewLine

End If

End If

If txtAddress.Text = "" Then Message = Message & "An Address" & vbNewLine 'Validations to check text boxes contain data.

If txtECName.Text = "" Then Message = Message & "An Emergency Contact Name" & vbNewLine

If txtECPhoneNumber.Text = "" Then 'Validating the Numbers entered are actually numbers

Message = Message & "An Emergency Contact Phone Number" & vbNewLine

ElseIf IsNumeric(txtECPhoneNumber.Text) = False Then

Message2 = Message2 & "The Emergency Phone Number isn't a number" & vbNewLine

ElseIf Len(txtECPhoneNumber.Text) > 13 Then

If Message3 = "" Then

Message3 = "The emergency phone number you entered is not the correct length" & vbNewLine & "A phone number must be 13 numbers with no spaces"

Else

Message3 = "A phone number must be 13 numbers with no spaces" & vbNewLine & "both phone numbers are the wrong length"

End If

End If

If Message <> "" Or Message2 <> "" Or Message3 <> "" Then 'If messages contain information some data has been entered incorrectly so won't be saved.

If Message2 = "" And Message3 = "" Then

MsgBox("Please Enter:" & vbNewLine & Message & vbNewLine & "This Employee has not been saved")

ElseIf Message = "" And Message3 = "" Then

MsgBox("Please Correct these mistakes:" & vbNewLine & Message2 & vbNewLine & "This Employee has not been saved")

ElseIf Message = "" And Message2 = "" Then

MsgBox(Message3)

Else

MsgBox("Please Enter:" & vbNewLine & Message & vbNewLine & "And Correct these mistakes:" & vbNewLine & Message2 & vbNewLine & "This Employee has not been saved" & vbNewLine & Message3)

End If

Else

If txtPassword.Text = "" Then 'Checking the Password has purposefully been left out.

If FirstUse = True Then

MsgBox("This employee must be an employee. Please enter a password." & vbNewLine & "This employee has not been saved")

Else

If MsgBox("Are you sure you don't want this employee to be an administrator?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

SaveData()

Else

MsgBox("Please enter a Password")

End If

End If

Else

If InputBox("Please re-enter the password") = txtPassword.Text Then 'Verifying the password entered is known by asking for it to be entered twice.

SaveData()

Else

If InputBox("The passwords didn't match please try again") = txtPassword.Text Then

SaveData()

Else

If InputBox("The passwords didn't match please try again") = txtPassword.Text Then

SaveData()

Else

MsgBox("Please Try again")

End If

End If

End If

End If

End If

End If

End Sub

Private Sub SaveData()

Dim RecordNumber As Short

Dim Save As Boolean = True

FileGet(IDFileNumber, IDRecord, EmployeeFileNumber)

If txtEmployeeID.Text <= IDRecord.ID Then 'Checking whether this is a new record or updating an old record.

If MsgBox("Are you sure you want to update this record, the old information will be lost." & MsgBoxStyle.YesNo) = True Then 'Checking with user they know they'll be overwriting old information.

For index = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord)

If txtEmployeeID.Text = EmployeeRecord.EmployeeID Then

RecordNumber = EmployeeRecord.RecordNumber

End If

Next

Else

Save = False

End If

Else

RecordNumber = LOF(EmployeeFileNumber) / Len(EmployeeRecord) + 1 'If not updating then creating a new unique identifier.

End If

If Save = True Then

EmployeeRecord.RecordNumber = RecordNumber 'Move data from form into a record

EmployeeRecord.EmployeeID = txtEmployeeID.Text

EmployeeRecord.EmployeeFirstName = txtFirstName.Text

EmployeeRecord.EmployeeLastName = txtLastName.Text

EmployeeRecord.JobTittle = txtJobTittle.Text

EmployeeRecord.PayRate = txtPayRate.Text

EmployeeRecord.PhoneNumber = txtPhoneNumber.Text

EmployeeRecord.Email = txtEmail.Text

EmployeeRecord.Address = txtAddress.Text

EmployeeRecord.EmergencyContactName = txtECName.Text

EmployeeRecord.EmergencyContactNumber = txtECPhoneNumber.Text

EmployeeRecord.Password = txtPassword.Text

FilePut(EmployeeFileNumber, EmployeeRecord, RecordNumber) 'Save the record in the file

If txtEmployeeID.Text = IDRecord.ID + 1 Then

IDRecord.FileNumber = EmployeeFileNumber 'Update the ID file so the last used ID is saved and therefore will not be used again.

IDRecord.ID = txtEmployeeID.Text

FilePut(IDFileNumber, IDRecord, EmployeeFileNumber)

End If

End If

End Sub

Private Sub CreateReport()

Dim index As Integer

lstEmployees.Items.Clear()

lstEmployees.Items.Add("Employee Employee Phone Email Address Job Pay Emergency Emergency ")

lstEmployees.Items.Add("ID Name Number Address Tittle Rate Contact Name Contact Number")

lstEmployees.Items.Add("========================================================================================================================================================================")

If LOF(EmployeeFileNumber) / Len(EmployeeRecord) = 0 Then 'If there's no employee record then this will be displayed in the list box.

lstEmployees.Items.Add("")

lstEmployees.Items.Add(" ---- No Employees ----")

Else

For index = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord) 'If there's records Cycle through file entering each record as a seperate line.

FileGet(EmployeeFileNumber, EmployeeRecord, index)

lstEmployees.Items.Add(String.Format(EmployeeFormat, EmployeeRecord.EmployeeID.ToString("D3"), RTrim(EmployeeRecord.EmployeeFirstName) & " " & RTrim(EmployeeRecord.EmployeeLastName), EmployeeRecord.PhoneNumber, EmployeeRecord.Email, EmployeeRecord.Address, EmployeeRecord.JobTittle, Format(EmployeeRecord.PayRate, "Currency"), EmployeeRecord.EmergencyContactName, EmployeeRecord.EmergencyContactNumber))

Next index

End If

End Sub

Private Sub btnDeleteRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDeleteRecord.Click

Dim index As Integer

Dim CurrentRecord As Integer = 1

Dim NumberOfRecords As Integer = LOF(EmployeeFIleNumber) / Len(EmployeeRecord)

Dim Temp As String = CurDir() & "\Temp.Dat"

Dim Deleted As Boolean = False

If MsgBox("Are you sure you want to delete the employee records for " & Trim(txtFirstName.Text) & " " & Trim(txtLastName.Text) & "?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

FileOpen(10, Temp, OpenMode.Random, , , Len(EmployeeRecord)) 'Create a temporary file

For index = 1 To NumberOfRecords

FileGet(EmployeeFIleNumber, EmployeeRecord, index)

If EmployeeRecord.EmployeeID <> txtEmployeeID.Text Then 'Fill file with every record wi=hich isn't being deleted. (Leaving out the one wanting to be deleted)

EmployeeRecord.RecordNumber = CurrentRecord

FilePut(10, EmployeeRecord, CurrentRecord)

CurrentRecord = CurrentRecord + 1

Else

Deleted = True

End If

Next index

If Deleted = True Then

FileClose(EmployeeFileNumber)

FileClose(10)

Kill(EmployeeFilePath) 'Delete the old file

FileCopy(Temp, EmployeeFilePath) 'copy temporary file to the old files name

Kill(Temp) 'Delete Temporary File

FileOpen(EmployeeFileNumber, EmployeeFilePath, OpenMode.Random, , , Len(EmployeeRecord))

FileOpen(DeletedEmployeeIDFileNumber, DeletedEmployeeIDFilePath, OpenMode.Append) 'Deleted Employee records IDs are saved in a seperate file to make it easier to make reports.

DeletedEmployeeIDRecord = txtEmployeeID.Text

PrintLine(DeletedEmployeeIDFileNumber, DeletedEmployeeIDRecord)

FileClose(DeletedEmployeeIDFileNumber)

Else

MsgBox("Sorry the Employee you tried to delete hasn't been deleted due to the Employee not existing")

End If

txtEmployeeID.Text = "" 'Data entry boxes are cleared as the employee has now been deleted.

txtFirstName.Text = ""

txtLastName.Text = ""

txtJobTittle.Text = ""

txtPayRate.Text = ""

txtPhoneNumber.Text = ""

txtEmail.Text = ""

txtAddress.Text = ""

txtECName.Text = ""

txtECPhoneNumber.Text = ""

txtPassword.Text = ""

CreateReport() 'Call to update the list box without the deleted file.

End If

End Sub

Private Sub lstEmployees\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles lstEmployees.SelectedIndexChanged

Dim RecordNumber As Short

RecordNumber = lstEmployees.SelectedIndex - 2 'This allows to click an item in the list box and the record number be found.

If RecordNumber < 1 Then

RecordNumber = 1

End If

ReadRecord(RecordNumber) 'This record number is passed to the ReadRecord procedure

End Sub

Private Sub ReadRecord(ByRef RecordNumber)

If LOF(EmployeeFileNumber) / Len(EmployeeRecord) > 0 Then

FileGet(EmployeeFileNumber, EmployeeRecord, RecordNumber) 'Open Record

RecordNumber = EmployeeRecord.RecordNumber 'Take data from record and place it in data boxes.

txtEmployeeID.Text = EmployeeRecord.EmployeeID

txtFirstName.Text = EmployeeRecord.EmployeeFirstName

txtLastName.Text = EmployeeRecord.EmployeeLastName

txtJobTittle.Text = EmployeeRecord.JobTittle

txtPayRate.Text = EmployeeRecord.PayRate

txtPhoneNumber.Text = EmployeeRecord.PhoneNumber

txtEmail.Text = EmployeeRecord.Email

txtAddress.Text = EmployeeRecord.Address

txtECName.Text = EmployeeRecord.EmergencyContactName

txtECPhoneNumber.Text = EmployeeRecord.EmergencyContactNumber

txtPassword.Text = RTrim(EmployeeRecord.Password)

End If

End Sub

Private Sub btnNextRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnNextRecord.Click

Dim RecordNumber As Short = 0

Dim NumberOfRecords, Index As Integer

NumberOfRecords = LOF(EmployeeFileNumber) / Len(EmployeeRecord)

If txtEmployeeID.Text = "" Then

RecordNumber = 1

Else

For Index = 1 To NumberOfRecords

FileGet(EmployeeFileNumber, EmployeeRecord, Index)

If EmployeeRecord.EmployeeID = txtEmployeeID.Text Then 'gets the record number of the next record

RecordNumber = EmployeeRecord.RecordNumber

If RecordNumber + 1 > NumberOfRecords Then

RecordNumber = NumberOfRecords

Else

RecordNumber = RecordNumber + 1

End If

End If

Next

End If

ReadRecord(RecordNumber) 'Passes the next record number to the ReadRecord procedure

End Sub

Private Sub btnLastRecord\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnLastRecord.Click

Dim RecordNumber As Short = 0

Dim NumberOfRecords, Index As Integer

NumberOfRecords = LOF(EmployeeFIleNumber) / Len(EmployeeRecord)

If txtEmployeeID.Text = "" Then

RecordNumber = NumberOfRecords

Else

For Index = 1 To NumberOfRecords

FileGet(EmployeeFIleNumber, EmployeeRecord, Index)

If EmployeeRecord.EmployeeID = txtEmployeeID.Text Then 'gets the record number of the next record

RecordNumber = EmployeeRecord.RecordNumber

If RecordNumber - 1 < 1 Then

RecordNumber = 1

Else

RecordNumber = RecordNumber - 1

End If

End If

Next

End If

ReadRecord(RecordNumber) 'Passes the next record number to the ReadRecord procedure

End Sub

Private Sub btnPrint\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPrint.Click

If (PrintDialog1.ShowDialog() = Windows.Forms.DialogResult.OK) Then 'Printing

PrintDocument1.DefaultPageSettings.Landscape = True

PrintDialog1.Document.Print()

End If

End Sub

Private Sub PrintDocument1\_PrintPage(ByVal sender As System.Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage

Dim yMargin As Integer = e.MarginBounds.Y

Dim xMargin As Integer = 10

Dim currentpageItemProgress As Integer = 0

For Each item As String In lstEmployees.Items

If (currentpageItemProgress >= ItemProgress) Then

e.Graphics.DrawString(item, lstEmployees.Font, New SolidBrush(lstEmployees.ForeColor), xMargin, yMargin)

yMargin += lstEmployees.Font.Size + 10

ItemProgress += 1

If (yMargin >= e.MarginBounds.Y + e.MarginBounds.Height And ItemProgress <= lstEmployees.Items.Count) Then

e.HasMorePages = True

currentpageItemProgress = 0

End If

End If

currentpageItemProgress += 1

Next

End Sub

Private Sub PrintDocument1\_EndPrint(ByVal sender As Object, ByVal e As System.Drawing.Printing.PrintEventArgs) Handles PrintDocument1.EndPrint

ItemProgress = 0

End Sub

Private Sub btnSearch\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSearch.Click

Dim Recordnumber As Integer = 0

If IsNumeric(txtSearch.Text) = True Then 'If user is searching form an employee by searching with an ID (Number)

For index = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord) 'For every employee record

FileGet(EmployeeFileNumber, EmployeeRecord, index)

If txtSearch.Text = EmployeeRecord.EmployeeID Then 'When ID is found save the records record number as RecordNumber

Recordnumber = EmployeeRecord.RecordNumber

End If

Next

If Recordnumber = 0 Then

MsgBox("Sorry the Id you entered could not be found" & vbNewLine & "The employee may have been deleted or never created") 'If searched and no ID found display an error message

Else

lstEmployees.SelectedIndex = Recordnumber + 2 'Set list boxes selected index as record number +2(3 lines of tittles) a procedure reacts to this change and loads the record.

End If

ElseIf IsNumeric(txtSearch.Text) = False And txtSearch.Text <> "" Then 'If employee is searched for by name

For index = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord) 'For every emplooyee record

FileGet(EmployeeFileNumber, EmployeeRecord, index)

If txtSearch.Text.ToLower.TrimEnd = EmployeeRecord.EmployeeFirstName.ToLower.TrimEnd Or txtSearch.Text.ToLower.TrimEnd = EmployeeRecord.EmployeeLastName.ToLower.TrimEnd Or txtSearch.Text.ToLower.TrimEnd = EmployeeRecord.EmployeeFirstName.ToLower.TrimEnd & " " & EmployeeRecord.EmployeeLastName.ToLower.TrimEnd Then

'Remove triling spaces and set all characters to lower case, compare firstname lastname and both names to the entered name

Recordnumber = EmployeeRecord.RecordNumber

End If

Next

If Recordnumber = 0 Then 'If record hasn't been found display error message

MsgBox("Sorry the Name you entered could not be found" & vbNewLine & "The employee may have been deleted or never created")

Else

lstEmployees.SelectedIndex = Recordnumber + 2 'If found isplay record

End If

Else

MsgBox("Please enter either an ID or employee name into the search box then try again") 'If the box is empty display an error message

End If

txtSearch.Text = ""

Private Sub btnClose\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClose.Click

Me.Close()

End Sub

End Class

## Setting/Help Form



Public Class frmSettingsHelp

Private Sub frmSettingsHelp\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

LoadMenuCombo() 'Call procedure to load combo box with menu names

End Sub

Private Sub btnMainMenu\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMainMenu.Click

Me.Close()

frmMenu.Show()

End Sub

Private Sub btnAdd\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAdd.Click

FileOpen(MenuFileNumber, MenuFilePath, OpenMode.Append) 'Open menu file (to add to the end)

MenuRecord = txtAddMenu.Text

PrintLine(MenuFileNumber, MenuRecord) 'Save data to menu file

FileClose(MenuFileNumber) 'Close record

txtAddMenu.Text = "" 'Set text box blank

txtAddMenu.Focus()

LoadMenuCombo() 'Call procedure to load menu combo box

End Sub

Private Sub LoadMenuCombo()

FileOpen(MenuFileNumber, MenuFilePath, OpenMode.Input) 'Open Menu file to read data from

cmbDeleteMenu.Items.Clear() 'clear comb box

Do While Not EOF(MenuFileNumber) 'For every line in the menu text file

cmbDeleteMenu.Items.Add(LineInput(MenuFileNumber)) 'Enter the line in the box

Loop

FileClose(MenuFileNumber) 'Close the menu file

End Sub

Private Sub btnDelete\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDelete.Click

Dim Temp As String = CurDir() & "Temp.Txt"

Dim Deleted As Boolean = False

Dim MenuToDelete, CurrentMenu As String

FileOpen(MenuFileNumber, MenuFilePath, OpenMode.Input) 'Open menu file

FileOpen(10, Temp, OpenMode.Output) 'Create temporary file

Do While Not EOF(MenuFileNumber) 'For every line in the menu file

MenuToDelete = cmbDeleteMenu.Text

CurrentMenu = LineInput(MenuFileNumber)

If MenuToDelete <> CurrentMenu Then 'If the line doen't equal the menu to be deleted

PrintLine(10, CurrentMenu) 'Save that line in the temporary file

Else

Deleted = True

End If

Loop

FileClose(MenuFileNumber) 'Close files

FileClose(10)

If Deleted = True Then

Kill(MenuFilePath) 'Delete menu file

FileCopy(Temp, MenuFilePath) 'Replace with temporary file

Kill(Temp) 'Delete temporary file

End If

cmbDeleteMenu.Text = ""

LoadMenuCombo()

End Sub

Private Sub btnBackUp\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnBackUp.Click

BackUp()

End Sub

Private Sub BackUp()

Dim BackUpFilePath As String

If (FolderBrowserDialog1.ShowDialog() = DialogResult.OK) Then

Dim TodaysDate As String = DateTime.Now.ToString("yyyy\_MM\_dd-hh\_mm")

BackUpFilePath = FolderBrowserDialog1.SelectedPath & "\" & TodaysDate

If (Not System.IO.Directory.Exists(BackUpFilePath & "\" & TodaysDate)) Then

System.IO.Directory.CreateDirectory(BackUpFilePath)

Else

Kill(BackUpFilePath & "\EmployeeFile.dat")

Kill(BackUpFilePath & "\IDFile.dat")

Kill(BackUpFilePath & "\MenuFile.txt")

Kill(BackUpFilePath & "\OrderFile.dat")

Kill(BackUpFilePath & "\OrderItemFile.dat")

If (System.IO.Directory.Exists(BackUpFilePath & "\DeletedEmployeeID.txt")) Then

Kill(BackUpFilePath & "\DeletedEmployeeID.txt")

End If

End If

FileClose(EmployeeFileNumber)

FileClose(OrderFileNumber)

FileClose(OrderItemFileNumber)

FileClose(EmployeeFileNumber)

FileClose(IDFileNumber)

FileClose(MealFileNumber)

FileCopy(EmployeeFilePath, (BackUpFilePath & "\EmployeeFile.dat"))

FileCopy(IDFilePath, (BackUpFilePath & "\IDFile.dat"))

FileCopy(MenuFilePath, (BackUpFilePath & "\MenuFile.txt"))

FileCopy(OrderFilePath, (BackUpFilePath & "\OrderFile.dat"))

FileCopy(OrderItemFilePath, (BackUpFilePath & "\OrderItemFile.dat"))

FileCopy(MealFilePath, (BackUpFilePath & "\MealFile.dat"))

If (System.IO.Directory.Exists(DeletedEmployeeIDFilePath)) Then

FileCopy(DeletedEmployeeIDFileNumber, BackUpFilePath & "\DeletedEmployeeID.txt")

End If

FileOpen(MealFileNumber, MealFilePath, OpenMode.Random, , , Len(MealRecord)) 'Open all files

FileOpen(OrderFileNumber, OrderFilePath, OpenMode.Random, , , Len(OrderRecord))

FileOpen(OrderItemFileNumber, OrderItemFilePath, OpenMode.Random, , , Len(OrderItemRecord))

FileOpen(EmployeeFileNumber, EmployeeFilePath, OpenMode.Random, , , Len(EmployeeRecord))

FileOpen(IDFileNumber, IDFilePath, OpenMode.Random, , , Len(IDRecord))

End If

End Sub

Private Sub btnUpload\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnUpload.Click

Dim UploadFilePath As String

If MsgBox("We suggest you backup the system before restoring it to an older date." & vbNewLine & "Would you like to backup first?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

BackUp()

End If

If (FolderBrowserDialog1.ShowDialog() = DialogResult.OK) Then

Dim NumberOfSlashes As Short

Dim Character As Char

Dim Last16Char As String = Microsoft.VisualBasic.Strings.Right(FolderBrowserDialog1.SelectedPath, 16)

For Each Character In Last10Char

If Character = "\_" Then

NumberOfSlashes = NumberOfSlashes + 1

End If

Next

If NumberOfSlashes = 3 Then

UploadFilePath = FolderBrowserDialog1.SelectedPath

FileClose(EmployeeFileNumber)

FileClose(OrderFileNumber)

FileClose(OrderItemFileNumber)

FileClose(EmployeeFileNumber)

FileClose(IDFileNumber)

FileClose(MealFileNumber)

FileCopy((UploadFilePath & "\EmployeeFile.dat"), EmployeeFilePath)

FileCopy((UploadFilePath & "\IDFile.dat"), IDFilePath)

FileCopy((UploadFilePath & "\MenuFile.txt"), MenuFilePath)

FileCopy((UploadFilePath & "\OrderFile.dat"), OrderFilePath)

FileCopy((UploadFilePath & "\OrderItemFile.dat"), OrderItemFilePath)

FileCopy((UploadFilePath & "\MealFile.dat"), MealFilePath)

If (System.IO.Directory.Exists(DeletedEmployeeIDFilePath)) Then

If (System.IO.Directory.Exists(UploadFilePath & "\DeletedEmployeeID.txt")) Then

FileCopy(UploadFilePath & "\DeletedEmployeeID.txt", DeletedEmployeeIDFileNumber)

Else

Kill(DeletedEmployeeIDFilePath)

End If

End If

FileOpen(MealFileNumber, MealFilePath, OpenMode.Random, , , Len(MealRecord)) 'Open all files

FileOpen(OrderFileNumber, OrderFilePath, OpenMode.Random, , , Len(OrderRecord))

FileOpen(OrderItemFileNumber, OrderItemFilePath, OpenMode.Random, , , Len(OrderItemRecord))

FileOpen(EmployeeFileNumber, EmployeeFilePath, OpenMode.Random, , , Len(EmployeeRecord))

FileOpen(IDFileNumber, IDFilePath, OpenMode.Random, , , Len(IDRecord))

LoadMenuCombo() 'Call procedure to load combo box with menu names

Else

MsgBox("Sorry these folders can't be uploaded as they are not the correct backedup files")

End If

End If

End Sub

Private Sub cmbDeleteFile\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbDeleteFile.SelectedIndexChanged

If MsgBox("We suggest you backup the system before deleting any files." & vbNewLine & "Would you like to backup first?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

BackUp()

End If

Select Case cmbDeleteFile.Text

Case "Menu"

DeleteMenu()

Case "Orders"

DeleteOrders()

Case "Orders & Employees"

DeleteOrders()

DeleteEmployees()

Case "Orders & Meals"

DeleteOrders()

DeleteMeals()

Case "All"

DeleteOrders()

DeleteEmployees()

DeleteMeals()

DeleteMenu()

End Select

End Sub

Private Sub DeleteOrders()

FileClose(OrderFileNumber)

FileClose(OrderItemFileNumber)

Kill(OrderFilePath)

Kill(OrderItemFilePath)

FileOpen(OrderFileNumber, OrderFilePath, OpenMode.Random, , , Len(OrderRecord))

FileOpen(OrderItemFileNumber, OrderItemFilePath, OpenMode.Random, , , Len(OrderItemRecord))

IDRecord.FileNumber = OrderFileNumber

IDRecord.ID = 0

FilePut(IDFileNumber, IDRecord, OrderFileNumber)

End Sub

Private Sub DeleteEmployees()

FileClose(EmployeeFileNumber)

Kill(EmployeeFilePath)

If (System.IO.Directory.Exists(DeletedEmployeeIDFilePath)) Then

Kill(DeletedEmployeeIDFilePath)

End If

FileOpen(EmployeeFileNumber, EmployeeFilePath, OpenMode.Random, , , Len(EmployeeRecord))

IDRecord.FileNumber = EmployeeFileNumber

IDRecord.ID = 0

FilePut(IDFileNumber, IDRecord, EmployeeFileNumber)

End Sub

Private Sub DeleteMeals()

FileClose(MealFileNumber)

Kill(MealFilePath)

FileOpen(MealFileNumber, MealFilePath, OpenMode.Random, , , Len(MealRecord))

IDRecord.FileNumber = MealFileNumber

IDRecord.ID = 0

FilePut(IDFileNumber, IDRecord, MealFileNumber)

End Sub

Private Sub DeleteMenu()

Kill(MenuFilePath)

FileOpen(MenuFileNumber, MenuFilePath, OpenMode.Append)

FileClose(MenuFileNumber)

End Sub

End Class

End Class

## Eastney Tavern Module

Module modEastneyTavern

Public CurrentEmployeeID As Short 'Login details stored to make ordering easier

Public CurrentUserAdministrator As Boolean

Public FirstUse As Boolean = False

Function GetEmployeeName() As String 'Function to return employee name from employee ID (signed in ID)

Dim index As Integer

For index = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord)

FileGet(EmployeeFileNumber, EmployeeRecord, index)

If EmployeeRecord.EmployeeID = CurrentEmployeeID Then

Return EmployeeRecord.EmployeeFirstName

End If

Next index

End Function

Public Structure Meal 'User defined data type created for meals

Dim RecordNumber As Short

Dim MealID As Short

<VBFixedString(20)> Dim MealName As String

<VBFixedString(15)> Dim Menu As String

Dim CostCurrent As Single

Dim PriceCurrent As Single

<VBFixedString(30)> Dim AllergyAdvice As String

Dim InStock As Boolean

Dim Historic As Boolean

End Structure

Public MealFormat As String = "{0,-5}{1,-21}{2,-16}{3,-7}{4,-7}{5,-6}{6,-9}{7,-30}" 'User defined format used to display the meal data in a list box.

Public MealRecord As Meal 'Create meal record

Public MealFilePath As String = CurDir() & "/MealFile.dat" 'Create meal file

Public MealFileNumber As Short = 1 'Assign an identifier for the file number (this makes it easier to code filehandling as you don't have to use numbers.

<VBFixedString(15)> Public MenuRecord As String 'Create Menu text file

Public MenuFilePath As String = CurDir() & "/MenuFile.txt"

Public MenuFileNumber As Short = 2

Public Structure Order 'Create user defined Order Record

Dim RecordNumber As Short

Dim OrderID As Integer

Dim TableNumber As Short

Dim OrderDate As Date

Dim EmployeeID As Short

Dim OrderCleared As Boolean

End Structure

Public OrderRecord As Order 'Create order File

Public OrderFilePath As String = CurDir() & "/OrderFile.dat"

Public OrderFileNumber As Short = 3

Public Structure OrderItem 'Create user defined order item record

Dim OrderID As Integer

Dim MealID As Short

Dim Quantity As Short

Dim Price As Single

Dim Cost As Single

End Structure

Public OrderItemRecord As OrderItem 'Create order Item File

Public OrderItemFilePath As String = CurDir() & "/OrderItemFile.dat"

Public OrderItemFileNumber As Short = 4

Public KitchenOrder As String = "{0,5}{1,20}" 'User defined formats for ordering form

Public Receipt As String = "{0,3}{1,15}{2,6}{3,6}"

Public Structure Employee 'Employee record user defined data type

Dim RecordNumber As Short

Dim EmployeeID As Short

<VBFixedString(15)> Dim EmployeeFirstName As String

<VBFixedString(15)> Dim EmployeeLastName As String

<VBFixedString(11)> Dim PhoneNumber As String

<VBFixedString(30)> Dim Email As String

<VBFixedString(30)> Dim Address As String

<VBFixedString(10)> Dim JobTittle As String

Dim PayRate As Single

<VBFixedString(15)> Dim EmergencyContactName As String

<VBFixedString(11)> Dim EmergencyContactNumber As String

<VBFixedString(15)> Dim Password As String

End Structure

Public EmployeeRecord As Employee 'Employee record

Public EmployeeFilePath As String = CurDir() & "/EmployeeFile.dat"

Public EmployeeFileNumber As Short = 5

Public EmployeeFormat As String = "{0,-9}{1,-32}{2,-12}{3,-31}{4,-31}{5,-11}{6,-6}{7,-16}{8,-14}"

Public DeletedEmployeeIDRecord As Short 'Record to store deleted Employee ID's (makes it easier to account for all sales in reports).

Public DeletedEmployeeIDFilePath As String = CurDir() & "/DeletedEmployeeID.txt"

Public DeletedEmployeeIDFileNumber As Short = 8

Public Structure ID 'File to store the highest used ID for every file (makes sure they're all unique.

Dim FileNumber As Short

Dim ID As Short

End Structure

Public IDRecord As ID

Public IDFilePath As String = CurDir() & "/IDFile.dat"

Public IDFileNumber As Short = 6

Public ItemProgress As Integer = 0 'Used for printing.

End Module

## Conversion Plan

Now the system has been completed I must decide how I will introduce the system into the pub. There are four main ways to change from the old system to the new and are as follows.

Parallel change over

This type of conversion involves installing and using the new system as well as using the old one. This can be useful for people to get used to how the new system works and once everyone is happy, the old system will be retired. This method is good that people can get used to the system in their own time, but the inconvenience of entering data into the computer and by hand is pointless and will cause errors through confusion.

Phased change over

This form of Change over isn’t really relevant to my project at the moment. It consists of introducing sections bit by bit eg. Introducing the orders form then if it works fine add on the analysis section. This may be useful in the future if any updates made.

Pilot run

A pilot run consists of trialing a system in one branch or department or a large company and if no problems occur then it will be rolled out throughout the whole organisation. This is obviously not relevant to a small independent chain like the Eastney Tavern, but would be a good way to trial a system if it grew into a chain.

Direct change over

This is the approach I will be taking; it involves installing the new system to the computer with appropriate printer and then using this system as the only system. This can be costly if the system breaks but due to the nature of my client they could easily revert back to the old system and as long as regular backups are taken no data should be lost.

# System testing

## Test Data

### Employee File

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | First name | Surname | Phone number | Email | Address | Job | Pay rate | EC Number | EC Name | Password |
| 1 | Chris | Moon | 07756827315 | Chris.Moon@outlook.com | 100 Cromwell Road | Owner | £00.00 | 0270015628 | Jax | Jet227 |
| 2 | Luke | Lindsey | 02392786651 | Luke.Lindsey@live.com | 17 Lance road | Chef | £7.80 |  | Mark |  |
| 3 | Darren | Munday | 07753210865 | Munday82@hotmail.co.uk | 222 Victoria grove | Head Chef | £12.80 | 0238768321 | Sarah | Cassarole3 |
| 4 | James | Legget | 07785921623 | James125@hotmail.com | 127 Short lane | Waiter | £6.60 | 0778562918721 | Lucy |  |
| 5 | Mark | Houghton | 07758632387 | HilseaHoughton@live.com | 32 Liverpool Boulevard | Waiter | £7.10 | 0239278641 | Janet |  |
| 6 | Louis | Smarse | 07727813453 | Lou.Smarse@gmail.com | 19 Short Lane | Waitress | £7.50 | 02365281328 | Jaime |  |

### Menu File

|  |
| --- |
| Starters |
| Entrées andappetizers |
| Traditional |
| Side Dishes |
| Desserts |
| Sunday lunch |

### Meal File

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Meal-ID | Meal Name | Menu | Cost | Price | Allergy Advice | In Stock | Historic |
| 1 | Satay Chicken Skewer | Starters | £0.80 | £3.95 | Egg & Nut | True | False |
| 2 | Fresh Seafood Cocktail | Starters | £1.10 | £4.95 | Egg, seafood | True | False |
| 3 | Seared Scallops Chorizo | Starters | £1.05 | £4.95 |  | True | False |
| 4 | Rib eye Steak | Traditional | £6.70 | £17.45 | Gluten Dairy | True | False |
| 5 | Portuguese Espetada | Traditional | £7.00 | £17.45 |  | True | False |
| 6 | Lamb Tagine | Traditional | £3.80 | £12.95 |  | True | False |
| 7 | Minted Lamb Burger | Traditional | £3.65 | £11.95 | Dairy  Gluten | True | False |
| 8 | Chips | Side Dishes | £0.30 | £2.45 | Gluten | True | False |
| 9 | Garlic Bread | Side Dishes | £0.30 | £2.45 | Gluten | True | False |
| 10 | Vegetables | Side Dishes | £0.20 | £1.95 |  | True | False |

### Order File

|  |  |  |  |
| --- | --- | --- | --- |
| Order-ID | Table Number | Date | Employee ID |
| 1 | 102 | 07/02/2014 | 1 |
| 2 | 204 | 15/03/2014 | 5 |
| 3 | 101 | 23/05/2014 | 4 |
| 4 | 301 | 24/05/2014 | 5 |
| 5 | 103 | 15/07/2014 | 1 |
| 6 | 210 | 21/09/2014 | 3 |
| 7 | 201 | 09/11/2014 | 6 |
| 8 | 302 | 15/01/2015 | 6 |

### OrderItem File

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order-ID | Meal-ID | Quantity | Price | Cost |
| 1 | 1 | 1 | £3.95 | £0.80 |
| 1 | 3 | 1 | £4.95 | £1.05 |
| 1 | 5 | 1 | £17.45 | £7.00 |
| 1 | 7 | 1 | £11.95 | £3.65 |
| 2 | 4 | 1 | £17.45 | £6.70 |
| 2 | 6 | 2 | £12.95 | £3.80 |
| 3 | 8 | 1 | £2.45 | £0.30 |
| 4 | 4 | 1 | £17.45 | £6.70 |
| 4 | 5 | 3 | £17.45 | £7.00 |
| 4 | 10 | 1 | £1.95 | £0.20 |
| 5 | 4 | 2 | £17.45 | £6.70 |
| 5 | 9 | 2 | £2.45 | £0.30 |
| 6 | 7 | 1 | £11.95 | £3.65 |
| 6 | 9 | 1 | £2.45 | £0.30 |
| 7 | 1 | 1 | £3.95 | £0.80 |
| 7 | 2 | 1 | £4.95 | £1.10 |
| 7 | 5 | 1 | £17.45 | £7.00 |
| 7 | 6 | 2 | £12.95 | £3.80 |
| 8 | 1 | 2 | £3.95 | £0.80 |
| 8 | 4 | 1 | £17.45 | £6.70 |
| 8 | 5 | 1 | £17.45 | £7.00 |
| 8 | 7 | 2 | £11.95 | £3.65 |

## Alpha testing

### Employee Form

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test number | Test data | | | Test type | | Test description | | | | | Expected outcome | | | | Actual outcome | | | | Corrections |
| 1 | 1st Employee test data | | | Normal | | Employees information is saved correctly | | | | | All inputs saved and displayed in the list box | | | | As expected | | | | None needed |
|  | | | | | | | | | | | | | | | | | | |
| 2 | 2nd row of employee data Without entering the first name | | Erroneous | | | | Testing that checks and finds the name is missing | | | | A message box will come up telling you the first name isn’t entered and the record hasn’t been saved | | | | As expected | | | | None needed |
|  | | | | | | | | | | | | | | | | | | |
| 3 | Full 2nd row of employee data | | Normal | | | | Test the system saves an employee correctly who isn’t an administrator (doesn’t have a password) | | | | The employee will be saved after a message box appears checking I don’t want him to be an administrator | | The message came up to check I didn’t want him to be an administrator but clicking both yes& no cause the same reaction, another message box asking for me to enter a password | | | | I have to change the code which says if the message box= true then save the employee, to If the message box =msgboxresult. yes  This should make the system work as expected  Pg.103 line 124 | | |
| If MsgBox("Are you sure you want to update this record, the old information will be lost.", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then 'Checking with user they know they'll be overwriting old information. | | | | | | | | | | | | | | | | | | |
| yes and No Both ask you to enter a password | | | | | | | | | | | | | | | | | | |
| 3.1 | Retest of 3 | | | | | | | | | | | | | As expected | | | | No more needed | |
|  | | | | | | | | | | | | | | | | | | |
| 4 | Record 3 from the employee test data, but with extra numbers 1 after the employee phone number | | | | Extreme | | | | Test to check phone number is not too long | | A error message will pop up and tell you the phone numer is too long, and the record won’t be saved | | | No error message was created and only the first 13 numbers were saved | | | | Need to add a validation check to make sure phone numbers are 13 Numbers. Updated employees form pg.101 line 53 | |
|  | | | | | | | | | | | | | | | | | | |
| ElseIf Len(txtPhoneNumber.Text) > 13 Then  Message3 = "The phone number you entered is not the correct length" & vbNewLine & "A phone number must be 13 numbers with no spaces"  End If | | | | | | | | | | | | | | | | | | |
| 4.1 | Record 4 but with a 1 at the end of the phone number | | | | Extreme | | | | Same as 4 | | Same as 4 | | | As expected | | | | No further corrections needed | |
|  | | | | | | | | | | | | | | | | | | |
| 5 | Record 4, saved first with a pay rate of “£5.50”. Then updated to “£6.60” | | | | Process testing | | | | Updating data | | The record will be saved and thn updated withno problems,This will be shwwn by the list box changeing | | | As expected | | | | No Corrections required | |
|  | | | | | | | | | | | | | | | | | | |
| 6 | Add Empployee Record 5 (add order and orderitem test data for later tesing) Delete test data 5 then add test data 6 | Process | | | | | | Testing deleting an employee, then adding a new employee after. | | Employee record is deleted and there’s no sign of it in the list box. When adding data item 6 the ID will be 6 and this will save normally | | As expected | | | | No corrections required | | | |
|  | | | | | | | | | | | | | | | | | | |

### Settings/Help

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Test data | Test type | Test description | Expected outcome | Actual outcome | Corrections |
| 1 | Menu test data 1 | Normal | Check the system saves a menu correctly | The menu will be saved and will appear as an option on the drop down (Delete menu) | As expected | None needed |
|  | | | | | |
| 2 | Menu test data 2 | Extreme | Check the system doesn’t save menus longer than 15characters | An error message will pop up and the menu won’t be saved | The whole menu was saved (all 22 characters) and dislayed in the combo box. This doesn’t crash the system but when entering meals and orders will only display the first 15 characters. | Add a validation to not allow menu names longer than 15 characters. And display an error message.  Changed code in Settinggs /Help pg.111 line 13-21 |
|  | | | | | |
| If Len(txtAddMenu.Text) <= 15 And txtAddMenu.Text <> "" Then 'Validation to check so menu is only saved with 15 or less characters and atleast 1  FileOpen(MenuFileNumber, MenuFilePath, OpenMode.Append) 'Open menu file (to add to the end)  MenuRecord = txtAddMenu.Text  PrintLine(MenuFileNumber, MenuRecord) 'Save data to menu file  FileClose(MenuFileNumber) 'Close record  txtAddMenu.Text = "" 'Set text box blank  txtAddMenu.Focus()  LoadMenuCombo() 'Call procedure to load menu combo box  Else  MsgBox("The menu has to be shorter than 15 characters" & vbNewLine & "This menu has not been saved" & vbNewLine & "Please shorten the name and try again")  End If | | | | | |
| 2.1 | Appetizers and Entrées | Same as 2 | Same as 2 | Same as 2 | As expected | No further corrections needed |
|  | | | | | |
| 3 | Test Data 2 | Process | Delete the second row of menu test data | The “Entrées and appetizers” Menu will be deleted and only “Starters” will be left in the combobox | Worked As expected | I’m going to add a message box to confirm the deleteion of the menu.  Pg.111 line 36 |
|  | | | | | |
| If MsgBox("Are you sure you want to delete the '" & cmbDeleteMenu.Text & "' menu", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then | | | | | |
| 3.1 | Same as 3 | Same as 3 | Same as 3 | A check message appears making sure you want ot delete the menu | As expected | No furether corrections needed |
|  | | | | | |
| 4 | All data in the system | process | BackUp the system , Delete all files then restore it. | Backing Up a dialogue will appear asking me to select a destination folder, When deleteing the files a message should appear checking if I want to backup. Finally when restoring a message will check if I want to backUp before getting me to select the file. | As expected | No correction needed |
| Select desktop as destination folder  New Time staped folder created  Delete all files  No  Menu (and everything else) deleted  Click Restore  Select BackUp Folder  Menu (and everything else) recovered | | | | | |
| 3.1 | An empty untitled folder | Erroneous process | Try to restore from a random folder | An error message will appear | As expected | No updates required |
|  | | | | | |

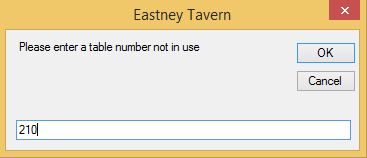
### Meal Form

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Test data | Test type | Test description | Expected outcome | Actual outcome | Notes |
| 1 | Record one from Meal file test data | Normal | Enter a normal selection of data and chrck it saves and displays correctly | The record willl be saved and displayed in the list box | As expected | No corrections needed |
|  | | | | | |
| 2 | Record 2 from meal test data | Extreme | Enter a meal with a meal name longer than 20 characters | An error message shoud come up nd the record won’t be saved. | No error message appeared and the first 20 characters of the meal name were saved | Add a validation check testing the length. Meal form pg.58 line 48 |
| If Len(txtMealName.Text) > 20 Then Message2 = "The meal name is longer than 20 characters." 'Check meal name is less than 20 char  If Len(txtAllergyAdvice.Text) > 30 Then Message2 = "The allergy advice is more than 30 characters" | | | | | |
| 2.1 | Record 3 | Same As 2 | Same as 2 | Same As 2 | As expected | No corrections needed (Record saved as “Scallops & Chroizo”) |
|  | | | | | |
| 3 | Record 4 with no name | Erroneous | Enter a record but without a meal name. | Error message telling me to enter a name and the record not being saved | As expected | No corrections needed (I saved the record correctly, with name) |
|  | | | | | |
| 4 | Record number 5, First saved as pice being &15.00 then updated to £17.45 | Normal | Enter a record incorrectly then update the record with the correct data. | The new data will be saved and displayed successfully | As expected | No corrections required |
|  | | | | | |

### Orders Form



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Test data | Test type | Test description | Expected outcome | Actual outcome | Notes |
| 1 | Order file 1 and order item file 1-4 | Normal | Enter a normal tables order. Login with the correct employee ID, | The data is saved and is displayed in the order and receipt. Also the total is found which should be £38.30 | As expected | No corrections needed. |
|  | | | | | |
| 2 | OrderID 2  But with quantity of 15 for the chips. | Extreme | Entering correct data but with a very large quantity | An error message will appear checking the quantity is correct | As expected | No corrections needed but the order was saved with 1 order of chips |
|  | | | | | |
| 3 | OrderID 3 | Process | Delete the order | A message will appear checking I want to delete the order then all the order related data will be deleted | All orders have been deleted which is obvious after exiting and enter ing the form | Correct code on page 81 line 361 |
|  | | | | | |
| Private Sub btnDeleteOrder\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDeleteOrder.Click  If MsgBox("are you sure you want to dlete this meal?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then  DeleteOrderItem()  DeleteOrder() 'Call procedure to delete order  txtOrderID.Text = "" 'Set input boxes to default  txtDate.Text = Today.Date  txtEmployeeName.Text = ""  txtTableNumber.Text = ""  LoadOpenOrders()  CreateReports()  End If End Sub  Sub DeleteOrderItem()  FileOpen(10, CurDir() & "/Temp.Dat", OpenMode.Random, , , Len(OrderItemRecord)) 'Open temporary file  For Index2 = 1 To LOF(OrderItemFileNumber) / Len(OrderItemRecord) 'For every order item record  FileGet(OrderItemFileNumber, OrderItemRecord, Index2)  If OrderItemRecord.OrderID <> txtOrderID.Text Then 'If order item isn't part of the order you want deleted  FilePut(10, OrderItemRecord) 'Then save record to temporary file  End If  Next  FileClose(OrderItemFileNumber) 'Close both files  FileClose(10)  Kill(OrderItemFilePath) 'Delete order item file  FileCopy(CurDir() & "/Temp.Dat", OrderItemFilePath) 'Copy temp file to order items location  Kill(CurDir() & "/Temp.Dat") 'Delete temp file  FileOpen(OrderItemFileNumber, OrderItemFilePath, OpenMode.Random, , , Len(OrderItemRecord)) 'Reopen ammended orderitem file  End Sub  Sub DeleteOrder()  Dim Index, OrderRecordNumber As Integer  OrderRecordNumber = 0  FileOpen(10, CurDir() & "/Temp.Dat", OpenMode.Random, , , Len(OrderRecord)) 'Create temporary file  For Index = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'For every order record  FileGet(OrderFileNumber, OrderRecord, Index)  If OrderRecord.OrderID <> txtOrderID.Text Then 'If the order record from file is not the one wanting to be deleted  OrderRecordNumber = OrderRecordNumber + 1  FilePut(10, OrderRecord) 'Save recor din temp file  End If  Next Index  FileClose(OrderFileNumber) 'Close both files  FileClose(10)  Kill(OrderFilePath) 'Delete order file  FileCopy(CurDir() & "/Temp.Dat", OrderFilePath) 'Copy temp file to order file location  Kill(CurDir() & "/Temp.Dat") 'Delete temp file  FileOpen(OrderFileNumber, OrderFilePath, OpenMode.Random, , , Len(OrderRecord)) 'Reopen ammended order file | | | | | |
| 3.1 | Same as test 3 | Same as test 3 | Same as test 3 | Same as test 3 | As expected |  |
|  | | | | | |
| 4 | OrderID 4 and paying with £80Cash | Process | Testing the system can calculate the change needed | The system will calculate the change as 15 - |  |  |
|  | | | | | |
| 5 | OrderID 5 but without a table number | Erroneous | Test the system doen’t try to save the order without a table number | The system will display azn error message asking me to enter a table number and the order won’t be saved | As expected | No correction required. OrderID was saved after with the correct table number |
|  | | | | | |
| 6 | Order ID 6 but with table number 102 | Process | New order made with a table number already in use by a different table.. | An error message will be displayed asking if I want to add the order to the current order on that table or add them to a new table |  |  |
|  | | | | | |

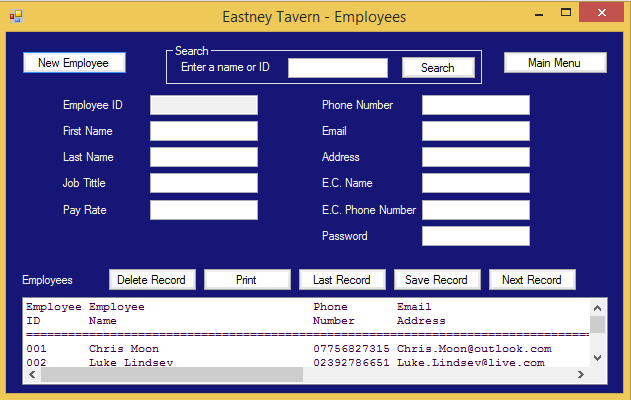


### Reports Form

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test number | Test data | Test type | Test description | Expected outcome | Actual outcome | Corrections |
| 1 | Employee IDs 1-5 OrderIDs 1-2 & 4-6  MealIDs 1 & 1-10. All ordered dates | process | Check data entered is what is given by the form. | Data to be tottalled with 18 meals, £134.95 profit and £207.6 Money taken | As expected | None needed |
|  | | | | | |
|  | | | | | |
| 2 | All Inputed info  Search name “mark” | Process | Search with on lowercase name | Mark Houghton will be displayed in the list box | As expected | No correction required |
|  | | | | | |
| 3 | Search Name “Bob” | Erroneous | Search incorrect name | The employee won’t be found and an error will be shown | As expected |  |
|  | | | | | |
| 4 | Employee ID 5 will be deleted, Employee ID 6 will be added and orders taken by him | Normal | Test Deleteing an employee that the name is removed and replaced by “unknown employee” | Mark Houghton wil be deleted fomt he report and an unknown emploee row will appear with; 2 orders, 8 meals and £115.1o money taken | As Expected | No crrections required |
|  | | | | | |
| 5 | Dates from 22/09/2014  to  08/11/2014 | erroneous | Between two dates that no orders were taken on dates | The system will display nought for every meal | As expected |  |
|  | | | | | |

### Navigation Test





With No Passsword

### Print Testing

Here I will test the printing processes inside the system, almost all the forms contain printing functionality. As I can’t print off these documents as part of this section please find them as labeled bellow in the Appendix (starting on page 257).

I will be printing with the current data inside the system, (all of test data without EmployeeID 5 and Order 3). For the receipt/orders print I will be printing off test Data with OrderID 5 (here I will print these off on A4 but they will be printed on receipt role at the pub. Reports will be using a date range including all orders.

The prints will be classed, as a success if they print off containing the correct information and they are formatted so all data is visible on each page

|  |  |  |  |
| --- | --- | --- | --- |
| Label (In appendix) | Document | Success | Notes |
| Fig. 1 | Meal file | True |  |
| Fig. 2 | Employee File | True | Just fits on a landscape page. |
| Fig. 3 | Order (OrderID 5) | True |  |
| Fig. 4 | Receipt (OrderID 5) | True |  |
| Fig. 5 | Employees (Alphabetical) | True | All these Reports are the same as are ordered the same in all 3 ways. |
| Fig. 6 | Employees (Tables Served) | True |
| Fig. 7 | Employees (Money Taken) | True |
| Fig. 8 | Employees (Search “Chris”) | True |  |
| Fig. 9 | Meals (Alphabetical) | True |  |
| Fig. 10 | Meals (By Popularity) | True |  |
| Fig. 11 | Meals (By Profit) | True |  |
| Fig. 12 | Meals (Search “Lamb”) | True |  |
| Fig. 13 | Gross Profits (Monthly) | True |  |
| Fig. 14 | Gross Profits (Quarterly) | True |  |

## Beta testing

After Completing my alpha testing I was happy the system was robust enough to allow the really users to try it out. I set it up on their system but as the printers haven’t been bought yet the system wasn’t used for actual service. However a lot of the future users have tried using the system and I have asked them to complete the following questionnaire.

**For waiting staff (without access to analysis)**

Job Tittle

What would you rate the current system out of 10

Do you think the order system is quicker to use then the old system?

Have you found the systems user interface easy to get used to?

Have you found any problems with the system (crashing, slow, etc.)?

Do you have any other comments on the system?

**For administrators**

Job Tittle

Do you find the login system easy to use?

Do you think the order system is quicker to use then the old system?

Have you found the systems user interface easy to get used to?

Have you found any problems with the system (crashing, slow, etc.)?

Do you think it’s easy to add employees and meals to the system?

Are there enough Reports and ways to order them? (If not what would you like added?)

Does this system give you more useful information than the old one?

Do you find the backup system easy to understand and use?

Generally what would you score the system out of 10?

Do you have any other comments on the system?

**Response 1**

**Job Title:** Waiter

**What would you rate the current system out of 10?**

8

**Do you think the order system is quicker to use then the old system?**

Definitely. It used to be a massive hassle to make an order, copying it out for the kitchen and again for a receipt. This system means all you have to do is press print. It is annoying having to select the printer every time you print, but this is still much quicker than copying it by hand.

**Have you found the systems user interface easy to get used to?**

Yes. It’s very simple, after Chris showed me what to do the first time I haven’t had any problems.

**Have you found any problems with the system (crashing, slow, etc.)?**

No.

**Do you have any other comments on the system?**

No.

**Actions**

No actions were needed as no problems were brought up in this response.

**Response 2**

**Job Title:** Waitress

**What would you rate the current system out of 10?**

6

**Do you think the order system is quicker to use then the old system?**

Yes, it’s much quicker and easier to create orders and receipts.

**Have you found the systems user interface easy to get used to?**

Yeah, it’s been fine.

**Have you found any problems with the system (crashing, slow, etc.)?**

No

**Do you have any other comments on the system?**

****I did accidentally click that an order had been cleared when it hadn’t. This was very annoying, as I had to ask the customer what they ordered and write out the receipt by hand.

**Actions**

I’ve added a dialogue so when the order cleared radio button is click it checks you definitely meant to click it.

**Response 3**

**Job Title:** Owner/Manager

**Do you find the login system easy to use?**

Yes, I think the login system is a great way to let orders be taken incredibly quickly and a good amount of security for important data. Although when first using the system I didn’t realise the password was case sensitive at first.

**Do you think the order system is quicker to use then the old system?**

Yes

**Have you found the systems user interface easy to get used to?**

Yes, it’s easy to use and the buttons are very obvious.

**Have you found any problems with the system (crashing, slow, etc.)?**

No

**Do you think it’s easy to add employees and meals to the system?**

Yes, it’s very easy and relatively instinctive. Although the first time I used the meals form I did find myself entering a menu and trying to write in a new meal at the top.

**Are there enough Reports and ways to order them? (If not what would you like added?)**

For the moment I think there is, in the future if I decide I do want drinks in this system as well there may be reason to get more reports analyzing how many drinks per table and even entering data of how many people to a table and at what time. But I there is definitely a nice amount at the moment.

**Does this system give you more useful information than the old one?**

Yes.

**Do you find the backup system easy to understand and use?**

Yes, it’s very nice. I didn’t even have to look at the manual, I like as I click on it the messages talk me through what to do.

**Generally what would you score the system out of 10?**

9

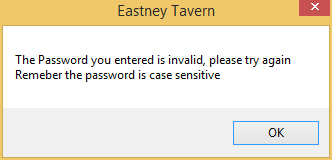
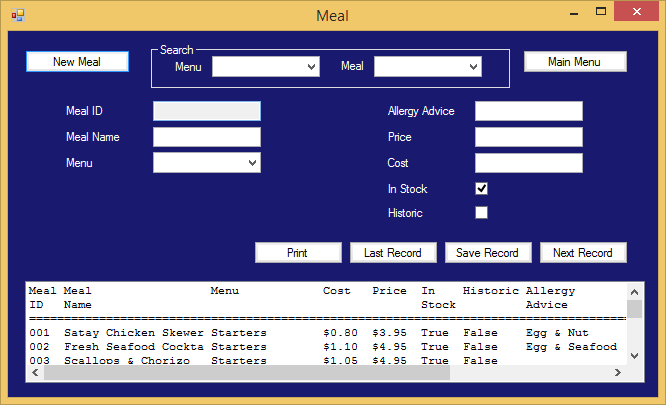
**Do you have any other comments on the system?**

No

**Actions**

I’ve added an extra line to the error box when an incorrect password is entered.

I have also moved the Menu and Meal search boxes into a box to more obviously distinguish between entering search data and new data.



## Acceptance testing

This section will reflect on the objectives set out in the analysis stage and check them against the current technical solution. As the client and users have now used the system I will sit down with the client to make sure he believes what I’ve created is what he wanted.

Sign off user manual

|  |  |  |
| --- | --- | --- |
| Objective | Met objective? | Comments |
| Qualitative objectives | | |
| The system should load quickly and run bug free. | True | No users have had a problem with waiting time. |
| The program should be very easy to use; it should have an intuitive user interface which is consistent throughout the system. The system should be simple enough that everyone no matter their computing skills can use the system. | True | The user interface has been easily adopted, with a few confusion now corrected (adding a box on the Meals form) |
| The system should have a login system, which stops unauthorized employees using certain aspects of the system. | True | The Client is very happy with the easy to understand, quick access security system. He was concerned about the time taken to login and this is exactly what he wanted. |
| The system should automatically calculate the price for each order. | True |  |
| The system should calculate the change needed when paying. | True |  |
| The program should produce many useful printed reports. | True | The client likes the current reports but is interested in adding extra functionality at a later date. |
| This project needs to be completed by Tuesday the 10th of February 2015. | False | This target has not been met. |
| Objective | Met objective? | Comments |
| Quantitative | | |
| The system must allow users to enter many meals on one order and the system will be able to automatically add up the total | True |  |
| The system must store records for at least six years for tax reasons | True |  |
| The system should print out the order information on two printers one as a receipt and the other as an order for the kitchen | True | The client would like the meals to be split up into menu type on the order. |
| The system should be easy to add important data about the meals such as; Meal-ID, MealName, Menu, Cost, Price, SupplierID, AllergyAdvice, InStock, Historic. This information will then be used to allow only the correct meals can be ordered during service and help keep a record of what’s selling | True | After a minor confusion of where to enter the menu when adding a new meal, this system has had an adjustment to correct that and no other problems have arisen. |
| The system must store detailed employee information which includes; Employee ID, EmployeeFirstName, EmployeeSurnameName, PhoneNumber, Email, Address, Job, PayRate, EmergencyContactName, EmergencyContactNumber. These will be used to contact employees for information regarding shifts and other stuff. It also has emergency contact information in case of an emergency | True |  |
| All the orders need to be entered in two files; these will include Order ID, Meal ID, Quantity, TableNumber, Date, EmployeeName | True |  |
| The system will also need to be able to print out complex reports showing how well various meals are selling | True |  |
| The system must run on the current hardware | True |  |

# System maintenance

## System overview

### System Development

At the beginning of this project I first had to understand how the current system works, what it’s used for and how it’s used. I did this in many ways, firstly I discussed at length the current system with the client (who’s the owner), I then observed the current system, I also asked some of the users to fill in a quick questionnaire, Finally I used some collected paperwork which I analysed to get a full and deep understanding of the current system and what people want and need in a new one. From this I decided on the data sources and data destinations, from here I created a preliminary data dictionary, which was used to create my volumetrics. From here data flow diagrams were created for both the current system and the proposed system as well as an e-r diagram for the future system. These were important to convey the new system to the client. I then created a list of quantitative and qualitative objectives, which were then checked and decided upon by the client, Now that I knew what the client wanted I decided how I would make it, I decided upon creating a bespoke system coded in Visual Basic .NET.

Now I knew what I needed to build I started to think how I would make it. I created an overall system design diagram and a modular structure diagram to help the client and I visualise the new system. I also updated the data dictionary to include all the necessary data types and information which the client doesn’t need to know, but is essential for a programmer. I then had to think about the main algorithms, which will be used to transfer the inputted data into useful information. As I did this I began to create mock ups of what the user interface would look like (I created a few and the client chose what he liked best). After creating a test plan to make sure the built program would run as expected I began to build the system.

After I built I realised I hadn’t included enough tests in my original test plan so added some more test data and plans as well as carrying out these tests and correcting any problems I found. After I was happy the system was fairly stable I set the system up on the clients computer and allowed the client and users to try adding their own data and using the system. This helped me get a sense of what needed to be added and improved. I finally went through the objectives set out at the beginning with the client and checked to make sure we were both happy I met them.

### System Structure

**Entity Relationship Diagram**

**Macintosh HD:Users:josephlaithwaite:Documents:College:A2:Computing:Comp 4 Coursework:Document:Entity Attribute Diagram 2.pdf**

**Modular representation of forms**

When building this system firstly I created all the forms I needed and coded the small amount of buttons needed to navigate between them. After this I Began to code individual forms, test these separately and then start on the next form. I started with the employees form as without this form you can’t login, Order things or create report. So once I could successfully add, save, update and delete employees I moved on to the login form.

The login form is where I open all the files and I made it so entering a valid employee ID and no password takes you to the Orders form (without access to the main menu) but if you enter a valid password as well you are sent to the Menu form. The menu form was already made, as this is only a form which closes itself and opens the form relating to the button.

Next I had to create part of the settings/Help form, here is where you add or delete menus (this had to be made before meals as it’s a foreign key in the meal record).

After this I could create the meal form, which allows you to save and edit meal records. Once this was working correctly it was time to create the orders form.

The orders form is the most important form of the system and will be the one used the most, this form is where orders are created and items are saved (without any input from the user) to two different files, the Order and OrderItem files. I made sure it was easy to use by only having two free entry data boxes (the table number and quantity) this will make it much harder to incorrectly enter data. Finally this form also allows you to delete a whole order or just one meal from the order.

Now all data capture forms have been created and the navigation around the system works well I produced the reports form. In this form you select a from date and a too date then select a button either Employee performance, meal performance or gross profit. These all create arrays and options form how to order them.

## A sample of detailed algorithm design

### Employee Search

This process takes entered text, which can either be an ID (number) or a first name, last name or full name (text). This algorithm then loops through the employee file comparing the entered data either to names or IDs and then changes the selected index in the of the list box to that of the record. This then calls a procedure to read the record.

Macintosh HD:Users:josephlaithwaite:Downloads:Blank Flowchart-4.pdf

**Actual Visual Basic .NET code, which is part of the Employees form.**

Private Sub btnSearch\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSearch.Click

Dim Recordnumber As Integer = 0

If IsNumeric(txtSearch.Text) = True Then

'If user is searching form an employee by searching with an ID (Number)

For index = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord) 'For every employee record

FileGet(EmployeeFileNumber, EmployeeRecord, index)

If txtSearch.Text = EmployeeRecord.EmployeeID Then

'When ID is found save the records record number as RecordNumber

Recordnumber = EmployeeRecord.RecordNumber

End If

Next

If Recordnumber = 0 Then

MsgBox("Sorry the Id you entered could not be found" & vbNewLine & \_

"The employee may have been deleted or never created")

'If searched and no ID found display an error message

Else

lstEmployees.SelectedIndex = Recordnumber + 2

'Set list boxes selected index as record number +2(3 lines of tittles) a procedure reacts to this

'change and loads the record.

End If

ElseIf IsNumeric(txtSearch.Text) = False And txtSearch.Text <> "" Then

'If employee is searched for by name

For index = 1 To LOF(EmployeeFileNumber) / Len(EmployeeRecord) 'For every emplooyee record

FileGet(EmployeeFileNumber, EmployeeRecord, index)

If txtSearch.Text.ToLower.TrimEnd = EmployeeRecord.EmployeeFirstName.ToLower.TrimEnd Or \_

txtSearch.Text.ToLower.TrimEnd = EmployeeRecord.EmployeeLastName.ToLower.TrimEnd Or \_

txtSearch.Text.ToLower.TrimEnd = EmployeeRecord.EmployeeFirstName.ToLower.TrimEnd \_

& " " & EmployeeRecord.EmployeeLastName.ToLower.TrimEnd Then

'Remove triling spaces and set all characters to lower case, compare firstname lastname and

'both names to the entered name

Recordnumber = EmployeeRecord.RecordNumber

End If

Next

If Recordnumber = 0 Then 'If record hasn't been found display error message

MsgBox("Sorry the Name you entered could not be found" & vbNewLine & \_

"The employee may have been deleted or never created")

Else

lstEmployees.SelectedIndex = Recordnumber + 2 'If found isplay record

End If

Else

MsgBox("Please enter either an ID or employee name into the search box then try again")

'If the box is empty display an error message

End If

txtSearch.Text = ""

End Sub

### Delete Order

This process is used to delete an order. This removes all the data for the relevant OrderID in both the Order and OrderItem file. It works by saving the orders and OrderItems, which don’t have the specified OrderID to a temporary file that then replaces the existing files.

**Flow Chart representation of the code.**

Macintosh HD:Users:josephlaithwaite:Downloads:Reports.pdf

**Actual Visual Basic Code from the orders form**

Private Sub btnDeleteOrder\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles \_

btnDeleteOrder.Click

If MsgBox("Are you sure you want to delete this order?", MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then

DeleteOrderItem()

DeleteOrder() 'Call procedure to delete order

txtOrderID.Text = "" 'Set input boxes to default

txtDate.Text = Today.Date

txtEmployeeName.Text = ""

txtTableNumber.Text = ""

LoadOpenOrders()

CreateReports()

End If

End Sub

Sub DeleteOrderItem()

FileOpen(10, CurDir() & "/Temp.Dat", OpenMode.Random, , , Len(OrderItemRecord)) 'Open temporary file

For Index2 = 1 To LOF(OrderItemFileNumber) / Len(OrderItemRecord) 'For every order item record

FileGet(OrderItemFileNumber, OrderItemRecord, Index2)

If OrderItemRecord.OrderID <> txtOrderID.Text Then

'If order item isn't part of the order you want deleted

FilePut(10, OrderItemRecord) 'Then save record to temporary file

End If

Next

FileClose(OrderItemFileNumber) 'Close both files

FileClose(10)

Kill(OrderItemFilePath) 'Delete order item file

FileCopy(CurDir() & "/Temp.Dat", OrderItemFilePath) 'Copy temp file to order items location

Kill(CurDir() & "/Temp.Dat") 'Delete temp file

FileOpen(OrderItemFileNumber, OrderItemFilePath, OpenMode.Random, , , Len(OrderItemRecord))

'Reopen ammended orderitem file

End Sub

Sub DeleteOrder()

Dim Index, OrderRecordNumber As Integer

OrderRecordNumber = 0

FileOpen(10, CurDir() & "/Temp.Dat", OpenMode.Random, , , Len(OrderRecord)) 'Create temporary file

For Index = 1 To LOF(OrderFileNumber) / Len(OrderRecord) 'For every order record

FileGet(OrderFileNumber, OrderRecord, Index)

If OrderRecord.OrderID <> txtOrderID.Text Then

'If the order record from file is not the one wanting to be deleted

OrderRecordNumber = OrderRecordNumber + 1

FilePut(10, OrderRecord) 'Save recor din temp file

End If

Next Index

FileClose(OrderFileNumber) 'Close both files

FileClose(10)

Kill(OrderFilePath) 'Delete order file

FileCopy(CurDir() & "/Temp.Dat", OrderFilePath) 'Copy temp file to order file location

Kill(CurDir() & "/Temp.Dat") 'Delete temp file

FileOpen(OrderFileNumber, OrderFilePath, OpenMode.Random, , , Len(OrderRecord))

'Reopen ammended order file

End Sub

## Procedure and variable descriptions

Here I will list all procedures and global variable. Local variables can be found by looking at the complete technical solution as entered earlier.

### Eastney Tavern Module

**Meal Structure**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| Record Number | Short | Stores the position in the file the record is stored in. |
| MealID | Short | The unique numerical identifier for the meal record. |
| MealName | String (20 characters) | The Name of the meal. |
| Menu | String (15 characters) | The name of the menu the meal is part of. |
| CostCurrent | Single | The cost of the meal to make at the current time. |
| PriceCurrent | Single | The price of the meal to buy as a customer at the current time. |
| AllergyAdvice | String (30 characters) | Contains any information about ingredients which cause common allergies which are in the meal |
| InStock | Boolean | Holds a value true or false whether the meal is in stock or not. |
| Historic | Boolean | Holds a value true or false if the meal is currently on the menu or not. |

**Order Structure**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| RecordNumber | Short | Holds the position of the record in the file. |
| OrderID | Short | The unique numerical identifier for the order. |
| TableNumber | Short | Stores the table number of where the orders been taken. |
| OrderDate | Date | The date the order is taken on. |
| EmployeeID | Short | The employee ID of whoever took the order |
| OrderCleared | Boolean | A yes/No variable which states if the table is in use or if the order has finished and the table is cleared. |

**OrderItem structure**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| OrderID | Short | An identifier from the order file used to link many meals to many orders |
| MealID | Short | An identifier from the meal file used to reference a meals data without having to duplicate it. |
| Quantity | Short | A value for how many of one meal is ordered in one order. |
| Price | Single | The price of the meal at the time it was ordered |
| Cost | Single | The cost of the meal at the time it was ordered |

**Employee Structure**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| Record Number | Short | Holds the position of the employee record in the file. |
| EmployeeID | Short | The unique numerical identifier for the employee |
| EmployeeFirstName | String (15 Characters) | Stores 15 characters of a first name |
| EmployeeLastName | String (15 Characters) | Stores 15 characters of a last name |
| PhoneNumber | String (11 Characters) | Holds an 11-digit phone number so the administrators can contact staff quickly. |
| Email | String (30 Characters) | 30 character email address which must include “@” symbol. Used for contacting employees. |
| Address | String (30 Characters) | Home address used to post pay slips. |
| JobTittle | String (10 Characters) | Contains the name of the employees’ job. |
| PayRate | Single | Hourly rate of pay |
| EmergencyContactName | String (15 Characters) | In case of emergency an emergency contact name is kept |
| EmergencyContactNumber | String (11 Characters) | And the emergency contacts 11-digit number |
| Password | String (15 Characters) | This is only used by administrators and stores a 15 character password which allows them passed just the order form. |

**ID Structure**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| FileNumber | Short | Holds the file number related to the ID record it’s holding. |
| ID | Short | Stores the Highest used ID of a file. |

**File Reference variables**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| MealRecord | Meal | The public variable for the meal structure |
| MealFilePath | String | The Directory information for the meal file |
| MealFileNumber | Short | Stores a number used to reference the meal file (without reusing and confusing numbers) |
| OrderRecord | Order | The public variable for the Order structure |
| OrderFilePath | String | The Directory information for the Order file |
| OrderFileNumber | Short | Stores a number used to reference the order file |
| OrderItemRecord | OrderItem | The public variable for the OrderItem structure |
| OrderItemFilePath | String | The Directory information for the OrderItem file |
| OrderItemFileNumber | Short | Stores a number used to reference the OrderItem file |
| IDRecord | ID | The public variable for the ID structure |
| IDFilePath | String | The Directory information for the ID file |
| IDFIleNumber | Short | Stores a number used to reference the ID file |

**Other Global Variables**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| MealFormat | String | The self-defined format to display a meal record correctly |
| KitchenOrder | String | A self-defined format to display an order for use in the kitchen |
| Receipt | String | A self-defined format to display an order as a receipt for customers |
| EmployeeFormat | String | A self-defined format to display an employee record correctly |
| CurrentEmployeeID | Short | Stores the ID of the employee who’s currently logged in. (stopping them from having to enter it again) |
| CurrentUserAdministrator | Boolean | If the current user logged in using a valid password this variable stores true. If false it is used to only show the orders form. |
| FirstUse | Boolean | This holds true if there are no employees and doesn’t let you in the system until you’ve created an administrator. |
| MenuFilePath | String | Holds the directory information for the menu file. |
| MenuRecord | String (15 characters) | Holds a menu name |
| MenuFileNumber | Short | Stores a number used to reference the Menu file |
| DeletedEmployeeID Record | Short | Holds The ID of a deleted employee until it’s saved in a file |
| DeletedEmployeeIDFile Path | String | Holds the directory information for the DeletedEmployeeID file. |
| DeletedEmployeeIDFile Number | Short | Stores a number used to reference the DeletedEmployeeID file |
| ItemProgress | Integer | Used for printing |

**Procedures**

|  |  |  |  |
| --- | --- | --- | --- |
| **Access** | **Procedure Name** | **Type** | **Description** |
| Public | GetEmployeeName | Function | Loops through the employee file to find the Current employee ID then passes the first name back. |

### Login Form

No global variables used in this form

**Procedures**

|  |  |
| --- | --- |
| **Procedure Name** | **Description** |
| frmLogin\_Load | Opens all files (except menu file and DeletedEMployeeID file), sets up ID file and specify if this is the first time used. |
| btnEnter\_Click | If the inputted employee ID has no password, it hides the login form and opens the orders form (with a logout button instead of main menu). If it has a valid password hides this form and opens the menu form. |
| ValidID  Function as boolean | Called to validate the ID entered exists, returning the function as true and finds the employee record number |
| ValidPassword  Function as boolean | Called to validate the password is correct if there is one by looking at the record number and returns true if this is what was entered. |
| btnExit\_Click | Ends the program |

### Menu Form

There are no global variables in this form

**Procedures**

|  |  |
| --- | --- |
| **Procedure Name** | **Description** |
| btnOrders\_Click | Hides this form and opens the orders form |
| btnMeals\_Click | Hides this form and opens the Meals form |
| btnEmployees\_Click | Hides this form and opens the Employees form |
| btnReports\_Click | Hides this form and opens the Reports form |
| btnSettingsHelp\_Click | Hides this form and opens the Settings/Help form |
| btnLogOut\_Click | Hides this form and opens the Login form |
| btnExit\_Click | Quits every form and ends the program |

### Meal For

There are no global variables in this form

**Procedures**

|  |  |
| --- | --- |
| **Procedure Name** | **Description** |
| frmMeal\_Load | Calls rcedures to Load a combo box and crreate a report |
| LoadMenuCombo | Opens the menu file and reads every line copying the text to the combo box. |
| btnMainMenu\_Click | Closes this form and loads the menu form. |
| btnNewMeal\_Click | Gets the ID record for the meal file and writes the ID +1 in the ID box. All other data entry boxes are emptied. |
| btnSaveRecord\_Click | Calls procedures to validate and save the record aswell as creating a report. |
| ValidateData | Checks every data entry box contains data, checks this data is within the correct length of characters, check it’s a number (if it’s supposed to be). Displays error messages if rquired, otherwise calls save data procedure. |
| SaveData | Checks if this record has been saved before, if it has it makes sure you want to update it then finds the record number for it. Otherwise it wrks out the record number as the last record number lus one. Then it saves the inputted data to the file. Finally if the ID is new, the file is opened and the highest used ID is updated. |
| CreateReport | Creates report first by writing the table headings then looping through the meal file displaying each record as one line (using custom format). |
| lstMeals\_SelectedIndexChanged | If you click a line in the list box this calculates the record ID of the record and passes this to the Read record procedure. |
| ReadRecord (ByRef RecordNumber) | Gets the record for the given record ID, enters it’s data into the data boxes and selecs the record in the list box. |
| cmbMenuSelector\_SelectedIndexChanged | This reads what menu has been selected in the combo box, then populates the meal file with every meal in that menu. |
| cmbMeal\_SelectedIndexChanged | Finds the meal record number by comparing every meal record with the meal in the list box. Thn passes this record number to the read record procedure. |
| btnNextRecord\_Click | Works out the current record number then passes this plus one to the read record procedure. |
| btnLastRecord\_Click | Works out the current record number then passes this minus one to the read record procedure. |
| btnPrint\_Click | Shows print dialogue and if this is used correctly prints the document. |
| PrintDocument1\_PrintPage | States the start margins, font size, colour and iterates the item progress by one for every line in the list box. |
| PrintDocument1\_EndPrint | When the print ends the ItemProgress variable is set to zero. |

### Employee Form

No global variables are used in this form

|  |  |
| --- | --- |
| **Procedure Name** | **Description** |
| frmEmployees\_Load | Calls pocedure to create a report and if this is the first time used, hides all buttons except new record save and shows a close button. |
| btnMainMenu\_Click | Closes this form and shows the menu form. |
| btnNewEmployee\_Click | Sets all dat input boxes blank excet The employee ID which it enters by reading the ID record and adding 1. |
| btnSaveRecord\_Click | Calls procedures to validate and save the record aswell as creating a report. |
| ValidateData | Checks every data entry box contains data, checks this data is within the correct length of characters, check it’s a number (if it’s supposed to be) and makes sure emails contain “@” signs. Displays error messages if rquired, otherwise calls save data procedure. |
| SaveData | Checks if this record has been saved before, if it has it makes sure you want to update it then finds the record number for it. Otherwise it wroks out the record number as the last record number plus one. Then it saves the inputted data to the file. Finally if the ID is new, the file is opened and the highest used ID is updated. |
| CreateReport | Creates report first by writing the table headings then looping through the employee file displaying each record as one line (using custom format). |
| btnDeleteRecord\_Click | Loops through the employee file and if the Employee ID isn’t the one wanting to be deleted then copy that record to a temporary file. Delete the employee file, copy the temporary file to the emloyeefilepath, delete the temporary file. Finally save the deleted ID to the DeletedEmployeeID file and empty the data boxes. |
| lstEmployees\_SelectedIndexChanged | If you click a line in the list box this calculates the record ID of the record and passes this to the Read record procedure. |
| ReadRecord(ByRef RecordNumber) | Gets the record for the given record ID, enters it’s data into the data boxes and selecs the record in the list box. |
| btnNextRecord\_Click | Works out the current record number then passes this plus one to the read record procedure. |
| btnLastRecord\_Click | Works out the current record number then passes this minus one to the read record procedure. |
| btnPrint\_Click | Shows print dialogue and if this is used correctly prints the document. |
| PrintDocument1\_PrintPage | States the start margins, font size, colour and iterates the item progress by one for every line in the list box. |
| PrintDocument1\_EndPrint | When the print ends the ItemProgress variable is set to zero. |
| btnSearch\_Click | Decides if an ID or name has been used to search. If it’s an ID, loop through the employee file and find the record number, select this item in the list box (calling the lstEmployees\_SelectedIndexChanged procedure). If it’s a name it does the same thing but compares names instead of ID’s. |
| btnClose\_Click | For first time users closes the employee form. |

### Orders Form

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| Total | Single | Holds the total price of an order, used by the CreateReports procedure and the btnPay\_Click |

|  |  |
| --- | --- |
| **Procedure Name** | **Description** |
| frmOrders\_Load | Calls procedures to load combo boxes and If the current user is an administrator (logged in wth a password) makes the main menu button visible and hides the logout button. |
| btnMainMenu\_Click | Hides this form and shows the menu form. |
| LoadOpenOrders | Loops throught he order file and if OrderCleared = false display the table number in the ComboBox. |
| cmbOpenOrders\_SelectedIndexChanged | Finds the order record with the same table number and hasn’t been cleared. Then it enters the ID, date, employe nam an sets the check boxes correctlt. Finally calls the create reports procedure. |
| btnNewOrder\_Click | Sets date as the current date, uses GetEmployeeName function to set employee name, gets the OrderID by adding one to the Id in the ID record. Finally sets the other boxes blank and calls a procedure to create reports. |
| MenuComboLoad | Opens the menu file and reads every line copying the text to the combo box. |
| cmbMenu\_SelectedIndexChanged | This reads what menu has been selected in the combo box, then populates the meal file with every meal in that menu. |
| btnSaveRecord\_Click | Calls procedures to validate and save the record aswell as Loading open orders. |
| ValidateData | Checks every data entry box contains data, checks this data is within the correct length of characters, check it’s a number (if it’s supposed to be) and makes sure emails contain “@” signs. Fills messages with error so if these aren’t empty it displas error messages, otherwise decides if this is the first meal in an order, if the order hasn’t been made calls procedure to Save the order. After this it alls a procedure to save the OrderItem. |
| SaveOrder | Saves the order informatiole and updated the ID file to have this ID as the highest ID. |
| SaveOrderItem | Get meal Record for seleted meal in the meal combo box, Saves the ID, pric and cost from this record with other data inputted to the ordr Item file. |
| CreateReports | Enter tittles on both list boxes, loop through all orderItems and if ID is the same as the ID in the text box, get the meal name from the meal record, display just quantity and name in order report and all the data on the receipt also for each order item add the price to the total, this is displayed alt the botom of the receipt. |
| rdbTableCleared\_CheckedChanged | If an order is selcted this finds the record the updates the checked status of the order saving it back again and calling a procedure to load open orders. |
| lstReceipt\_SelectedIndexChanged | Makes the selected index the same line in the order list box as the receipt list box. |
| lstOrder\_SelectedIndexChanged | Makes the selected index the same line in the receipt list box as the order list box. |
| btnDeleteMeal\_Click | If the selected line starts with a quantity, get the meal name from the line, find the meal ID for this meal. Copy every orderitem record into a temporary file except the orderitem with the same Meal ID and orderID. Delete the orderitem file, copy the temporary file to the orderitem directory finally delete the temporary file. |
| btnLogOut\_Click | Close this form, show the login form and set the current employeeID to zero. |
| btnDeleteOrder\_Click | Checks you want to delete the order then call a procedure to delete orderitem and one to delte order, load open orders and create a report. |
| DeleteOrderItem | Loops through orderitem file and saves all record without the same orderID (to be deleted) to a temporary file which then replaces the orderitem file. |
| DeleteOrder | Loops through order file and saves all the records without the orderID to be deleted to a temporary file which then replaces the order file. |
| btnPrintOrders\_Click | Shows print dialogue to print the order report and if this is used correctly prints the document. |
| PrintDocument1\_PrintPage | States the start margins, font size, colour and iterates the item progress by one for every line in the list box. |
| PrintDocument1\_EndPrint | When the print ends the ItemProgress variable is set to zero. |
| btnPrintReceipt\_Click | Shows print dialogue to print the receipt report and if this is used correctly prints the document. |
| PrintDocument2\_PrintPage | States the start margins, font size, colour and iterates the item progress by one for every line in the list box. |
| PrintDocument2\_EndPrint | When the print ends the ItemProgress variable is set to zero. |
| btnPay\_Click | Tells you how much change to give a customer after they pay with cash. |

### Reports Form

**MealStructure**

This structure is created to hold data, which will be displayed in a list box. The data is calculated then saved in a two-dimensional array using this structure. It can then be ordered and searched through much quicker.

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| MealName | String (20 Characters) | Holds the name of the meal. |
| NumberOfMeals | Short | Holds the total number of meals sold (during the time frame) |
| MoneyTaken | Single | Holds the total amount of money taken (during the time frame) |
| Profit | Single | Holds the total profit (during the time frame) |

**EmployeeStructure**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| EmployeeName | String (31 Characters) | Holds the full name of the employee. |
| NumberOfOrders | Short | Holds the total number of orders (tables) covered by this employee. (during the time frame) |
| NumberOfMeals | Short | Holds the total number of meals sold by this employee (during the time frame) |
| MoneyTaken | Single | Holds the total amount of money taken by this employee (during the time frame) |

**GrossProfit structure**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| Month | Short | Holds the month (eg. 1 for January) |
| Year | Short | Holds the year (eg. 2015) |
| MoneyTaken | Short | Holds the total amount of money taken in a month (during the time frame) |
| Profit | Single | Holds the total amount of Profit made in a month (during the time frame) |

**Other Global Variables**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| Order Exists | Boolean |  |
| Meals((LOF(MealFileNumber) / Len (EmployeeRecord) ) | MealStructure | An array to hold all the information about meal sales, which can then be ordered and searched quickly in any way. |
| Employees(LOF (EmployeeFileNumber) / Len(EmployeeRecord) + 1) | EmployeeStructure | An array to hold all the information about employees perfomance which can then be ordered and searched quickly in any way. |
| FormatEmployeeReport | String | A self-defined format to display employee performance (from the array) correctly. |
| FormatMealsReport | String | A self-defined format to display meal sales (from the array) correctly. |
| Profits() | GrossProffit | An empty array, which is redimed every time data is added. This will hold information about monthly profits and make it easy to combine for quarterly info. |
| NumberOfMonths | Integer | A variable to hold the number of months stored which is the length of the array. |
| FormatMonthlyProfit | String | A self-defined format to display gross profit info (from the array). |

**Procedures**

|  |  |
| --- | --- |
| **Procedure Name** | **Description** |
| frmReports\_Load | If no orders have been taken this will not allow you to enter any dates or create a report. Otherwise it sets up the date boxes from the irst order to the last order. |
| PopulateEmployeeArray | If employees have been deleted it loops through the DeletedEmployeeID file and then throught the orderitem file to total up all the information and place it in an “Unknown Employee” Record. It also does this with every employee file comparing the EMployeeID with an order record, getting that orderID then getting the orderITem information. These are all saved to the empliyees array. |
| PopluateMealArray | Loops through every order Item and gets the information from these for every mealID and saves this in the Meals array. |
| btnMainMenu\_Click | Closes this forma and opens the menu form. |
| btnEmployeePerformance\_Click | If orders haven’t been taken it displays an error message, otherwise it shows the ordering buttons and search bittons aswell as calling the PopulatEmployeeArray procedure. |
| DisplayMealList | Procedure to look at the Meals array and display the information using the custom format in the listbox. It also totals up everything and displays this at the end. |
| btnMealAlphabetical\_Click | This orders the Meals array in alphabetical order. It does this by looping through the array (number of records)2  times, comparing the last to the current record and swapping if necessary,using a temporary record. |
| btnMealsByPopularity\_Click | Same as btnMealAlphabetical\_Click but comparing NumberOfMeals instead of name |
| btnByProfit\_Click | Same as btnMealAlphabetical\_Click but comparing Profit instead of name |
| btnEmployeesAlphabetical\_Click | Same as btnMealAlphabetical\_Click but looking through the employees file. |
| btnTablesServed\_Click | Same as btnMealAlphabetical\_Click but looking through the employees file and comparing NumberOfOrders instead of Name. |
| btnMoneyTaken\_Click | Same as btnMealAlphabetical\_Click but looking through the employees file and comparing MoneyTaken instead of name. |
| DisplayEmployeeList | Procedure to look at the Employees array and display the information using the custom format in the listbox. It also totals up everything and displays this at the end. |
| btnMeals\_Click | If orders haven’t been taken it displays an error message, otherwise it shows the ordering buttons and search buttons aswell as calling the PopulatMealArray procedure. |
| btnMealSearch\_Click | Loops through the meal array and compares what you enetered to search with the full meal name (all in lowercase) and splits the meal name if there are spaces and compares these. If any matches are found then the array item is placed in the list box. |
| btnEmployeeSearch\_Click | Same as btnMealSearch\_Click but looking through the employee file |
| datFromDate\_ValueChanged | Makes all ordering and searching buttons disappear aswell as clearing the list box. |
| datToDate\_ValueChanged | Same as datFromDate\_ValueChanged |
| btnPrint\_Click | Shows print dialogue and if this is used correctly prints the document. |
| PrintDocument1\_PrintPage | States the start margins, font size, colour and iterates the item progress by one for every line in the list box. |
| PrintDocument1\_EndPrint | When the print ends the ItemProgress variable is set to zero. |
| btnGrossProfit\_Click | Clears list box, shows relevant ordering buttons and calls procedure to PopulateGrossProfitArray. |
| PopulateGrossProfitArray | Loops through order file and for every order taken in a given month in a given year then total these up, when the next month is found reset totals and save it to the GrossProfit Array. |
| btnMonthly\_Click | Loops through the GrossProfit Array and using a case statement finds the month name from the month number, it then adds the totals to the grand total and displays all the information using a custom format. |
| btnQarterly\_Click | Similar to btnQarterly\_Click bu uses the case statemnt to select the quarter (eg. 1,2,3 = first) |

### Settings/Help Form

|  |  |
| --- | --- |
| **Procedure Name** | **Description** |
| frmSettingsHelp\_Load | Calls procedure to LoadMenuCombo |
| btnMainMenu\_Click | Hides this form and shows the menu form. |
| btnAdd\_Click | Saves the text entered to the menu file. |
| LoadMenuCombo | Opens the menu file and loops through it copying every line to the combo box. |
| btnDelete\_Click | Copies every line form the current menu file to a temporary file except the menu item currently selected in the combo box. It then replaces the menu file with the temporary file. |
| btnBackUp\_Click | Calls the BackUp procedure |
| BackUp | Asks the user to input a destination folder, creates a folder directory from the date and time then copies the dataa files from the system to this external folder. |
| btnUpload\_Click | Firstly checks if you want to back up the systsem before restoring (if you do it calls the BackUp procedure). It then asks you to chose the backup folder wanted, checks this is a valid folder than deletes the current files in the system and copies the ones from this folder to the system. |
| cmbDeleteFile\_SelectedIndexChanged | Checks if you want to backUp the system first (if you do it call the BackUp procedure) otherwise it uses a case statement ot decide which procedure to call (most files have there own delete procedure). |
| DeleteOrders | Deletes the order file and the OrderItem file. |
| DeleteEmployees | Deletes the employees file and if the DeletedEmployeeID file exists deletes that too. |
| DeleteMeals | Deletes the meal files. |
| DeleteMenu | Dleetes the Menu File. |

## Annotated listings of program code/macro code and tailoring

Find these in the Technical solution section starting on page 61

# Appraisal

## Comparison of project performance against numbered general and specific objectives

In this section I will be comparing the objective originally set out in the analysis section with the finished system I have created.

### Qualitative Objectives

1. The system should load quickly and run bug free.

This system has met this target. By creating this system with a fast, bug free system in mind I’ve managed to integrate a large amount of validations into my system, this in itself minimises the possible bugs as no erroneous data can be entered so the system won’t break trying to read data in a way it can’t. I also tried to only search through files when vitally important in order to decrease the load time, which I believe is very fast.

1. The program should be very easy to use; it should have an intuitive user interface which is consistent throughout the system. The system should be simple enough that everyone no matter their computing skills can use the system.

I believe this target has been met; the colour scheme and general layout of the forms are consistent throughout. The meals and employees forms are practically the same form, with the reports and Settings/Help form keeping the same position of the “Main Menu”. This consistency therefore means you only need to learn to use one form and the others are completely natural. Learning to use the first form is very easy as buttons and labels are written in very clear, unambiguous text. I have also included an imbedded help form, which steps through every aspect of the system.

1. The system should have a login system, which allows most employees only access to the orders form and only allow administrators access to changing data and analytical tools.

I have met this target by allowing users to login to the orders form using only their Employee ID, most of the staff have no need for a password and can’t possibly see any of the financial information. This said it’s still very easy to login to gain full access, you simply must enter a password as well as the ID and you have full access to add, delete, edit and analyse everything.

1. The system should automatically calculate the price for each order.

This is done in the orders form, when a new meal has been added the receipt report is updated with the new meal as well as its price and adds this price to the total.

1. The system should calculate the change needed when paying.

I have allowed the user to do this in the orders form by clicking the “Pay” button. They can then enter the amount paid and t will calculate the amount of change required.

1. The program should produce many useful printed reports.

I believe I have done this well offering a good selection of reports analysing meal sales, employees’ orders and profits with a selection of ways to order this information (alphabetical, total etc.). These on screen reports can also be printed with ease. Of course more reports could be created from the data taken and I’ll be able to update the system to include extra reports if the client realizes that he wants something in the future.

1. The system should contain a lot of dropdown boxes to minimize the amount of free text entered, this is to reduce errors and speed up data entry

I believe this target was completed successfully. For example, the Orders form contains only two free text entry boxes; this speeds up using the system dramatically and reduces the number of possible errors.

1. This project needs to be completed by Tuesday the 10th of February 2015.

This objective has not been met. I believe I underestimated the amount of time needed to create a bespoke complex system to a high standard so overran my initial timeline by a few weeks.

### Quantitative Objectives

1. The system must allow users to enter many meals on one order and the system will be able to automatically add up the total.

I have managed to meet this objective by reloading the Receipt Report on the order form every time a new meal is added, this then looks for the meal record and adds up the total price as it goes.

1. The system must store records for at least six years for tax reasons.

The system has been designed to have no realistic limit on the number of orders (Actual limit = 2 billion order items so around ¼ billion orders), or meals and employees (which have an actual limit of 32,767) that the system can hold. If somehow the limits are reached then the integrated back up system can be used to store a copy of the current information then this can be deleted and started again.

1. The system should print out the order information on two printers one as a receipt and the other as an order for the kitchen.

I have met this target by displaying the two reports on the orders screen and giving the user the option of which report they want to print to which printer.

1. The system should be easy to add important data about the meals such as; Meal-ID, MealName, Menu, Cost, Price, SupplierID, AllergyAdvice, InStock, Historic. This information will then be used to allow only the correct meals can be ordered during service and help keep a record of what’s selling.

This objective has been met and can easily be done in the meals form.

1. The system must store detailed employee information which includes; Employee ID, EmployeeFirstName, EmployeeSurnameName, PhoneNumber, Email, Address, Job, PayRate, EmergencyContactName, EmergencyContactNumber. These will be used to contact employees for information regarding shifts and other stuff. It also has emergency contact information in case of an emergency.

This objective has been met and can easily be done in the employees form.

1. All the orders need to be entered in two files; these will include Order ID, Meal ID, Quantity, TableNumber, Date, and EmployeeName.

This objective has been completed so when you use the order form the only data you as a user have to enter is; table number, meal and quantity. The rest of the data is worked out and entered by the system and then saves this data fully normalised to the Order file and OrderItem file.

1. The system will also need to be able to print out complex reports showing how well various meals are selling.

This objective was met. This can be completed in the Reports form where you select a date range you want to see the data of, select whether you want to see “Meals”, “Employee performance” or “gross profit “ then select how you want to display the data. Now the on screen report has been created it can be printed to create a hard copy of the report.

1. The system must run on the current hardware

This target has been met as windows XP is the current hardware and the lowest requirement for the system. This system has also been designed with the future in mind and will work perfectly with any new Windows operating system up to Windows 8.

## User feedback authenticated by assessor

I sent an email containing a questionnaire to the client and here is his response, it contains the questions and their relevant answers. Find evidence in Appendix labeled Fig. 15.

**Are you happy with what I’ve created for you?**

Yes, the system is very good, it contains everything I wanted and works very nicely together.

**What would you rate the final system and all of relevant documentation out of ten?**

User Manual 7/10

System Maintenance 8/10

Computer system 9/10

**Do you think this system loads reasonable quickly?**

So far the system has loaded incredibly fast with every use. The login form is pretty much instantaneous and I haven’t had any problems with navigating through the form and having to wait for something to load.

**Have you found the user interface/Navigation easy to understand and use?**

Yes, the navigation works well, it allows me to very quickly move from one form to another entering data quickly and then reviewing these changes elsewhere. This interface is also very useful, as very little training is needed to teach new waiting staff.

**Do you think the login system provides adequate security whilst keeping speedy access?**

Yes, this login system is much better than expected. One of my major reluctances for a computer system was having either all the important monetary information open to all my employees or having a long and complicated login, which would be impractical on busy days. This therefore is great the fact that waiting staff don’t even have a password so can’t ever get in the system is great.

**Have you found the Orders form lets you enter many meals on one form and accurately calculates the total price and change needed for an order?**

Yes, the orders form allows me to add as many different meals to the same meal, I really like that I can come back and add more later. All the receipts I’ve had made are accurate, and the change function also works fine.

**Do you think the Reports are useful and the printing function easy to use?**

Yes, the reports will be very useful, as it automatically keeps track of all the money coming in, this will be great when doing the taxes and is very useful for me to keep track of how the pub’s doing month by month, the inclusions of drinks in the system would make this perfect though. The other reports of meal and employee performance are good, It may be nice to be able to look at meals by menu and employees by job tittle but it’s still very useful having all this information to tailor our menu to what the customer are buying.

**Do the records hold all the information you want (for Meals, employees & Orders)?**

Yes, they hold plenty of information, the employee records may even hold too much, all the information is very useful for sending pay slips to the correct address and letting people know their shifts, but it is a lot of information to take when someone’s only working 4 or 5 hours a week. But they do all hold exactly what I wanted.

**Do you understand the Backup system and do you think this is a useful addition to the original plan?**

Yes, I’m so pleased you managed to add this in. I was suppressed when I stumbled across it and found out how to use it just through experimenting, it’s very well signposted with the messages. I think this is great because that was another one of my big worries that all this information will be saved somewhere and through a random fault it could be lost. This really puts that worry to rest.

**Do you think the user manual (both the embedded one and the separate hard document) includes enough information on how to use the system?**

I think the User manual is very extensive, it a lot of information about every single feature on every slide and it’s easy to find what I’m looking for as it’s split into forms and then specific functions. I do like having the inbuilt help file but it would be much better for this to include the screen shots to help guide me through.

**Is there any extra functionality you think you’d like (Now or in the near future) to enhance the system?**

Having a separate form to enter drink, and a way to add drinks as orders, would be great. As I said this would fully complete the system so no extra income would have to be calculating anywhere else, therefore this would produce all the evidence I need for revenue and customs. Another thing that could be incorporated would be a clock in system. I think it would be great if somehow the employees working could log in, clock in or out and the total wages they needed can be worked out somewhere.

## Analysis of user feedback

Throughout the production of this system I have had almost constant communication with the client. This has helped me fully understand what I’m making and make sure it’s exactly what the client wants.

My first major communication with the client was during the analysis, here I conducted an interview with the client (the owner of the pub) and two users (the head chef and a waitress), this gave me key insight to how the current system worked and the many problems with it. From Understanding the current system I then heard what the users would like to see in a new system. I then developed a list of objectives of what I thought the new system should include, this was firmed up with the client and any extras/changes he wanted were made.

From here I had to start thinking exactly how I change the list of objectives into a fully functioning system. For the most part this was in too much detail for the client to understand, however when designing the user interface I created many different styles and got the client to decide on which design (or aspect) he likes and these formed the bases for my system moving into the building section. During my build there wasn’t a huge amount of client interaction as I felt I knew what he wanted and didn’t need too much guidance, this said I did show him the system in the early development process just to make sure he liked the interface.

Once the system was built I began to alpha test it, this brought it up to a standard I was happy with, and believed could be used without bugs or errors. I then could pass the system to the client for beta testing, here the client and users got hands on experience and fed back to me telling me what did and didn’t work well. For the most part the feedback was very positive with only a few minor problems that I fixed.

Finally I got the client to fill in a questionnaire asking him question relevant to the initial objectives, as well as asking if there is any further development he would like. This feedback confirmed that all objectives (except the deadline) were completed to a satisfactory standard as well as proposing a few features he thinks he may want in the future (I’ll go into more detail on further developments in the next section).

Overall the client was very happy with the system produced and thinks it will improve the efficiency of his pub massively.

## Possible extensions

Now I’ve completed the system I would like to outline future developments I think could work well, this will be a mix of new functionality coupled with some more advanced algorithms which could be used in place of the current, less efficient, ones.

Firstly a major improvement to the system would be adding a scheduler to the backup system. Being able to back up is great but when a user has never had to restore a system, they don’t understand its importance and therefore don’t back up the files as frequently as required. This is why adding a function to automatically backup the system hourly, daily or every time the system is quit would be great.

Another, more major, extension to the system would be creating a clock in/out process. This would work by making users sign in and clock in in a new form, the system would calculate how long you’ve worked before you clock out. The system could then calculate Daily/Weekly/Monthly hours worked and use this to calculate pay. This would also make the gross profit report much more accurate as could include more of the costs.

The system may also be upgraded to include drinks; this could be done using the current system and having drinks as a menu then enter them as a normal meal. I would prefer to create a different file for this and make the orders form calculate them separately (eg. Not sending drinks orders to the kitchen).

As the system is used the amount of data in the system will grow dramatically. So the system may have to be edited to increase the efficiency and speed of the program.

A large oversight of mine was using linear searches instead of binomial searches when looking through a file for ID’s (not names as lists aren’t alphabetically ordered). Instead of looking at every single record and comparing it with what you’re looking for I could change it to look at the middle record, compare this with what I’m looking for then if it’s not what I’m looking for split the list again (of the side with the ID on) and continue until the ID is found. This takes a lot fewer attempts and would speed up the system considerably. I could implement this on the meal for when searching if the meal ID already exists, navigating up and down the record and in the orders form checking if the order has already been created.

I would also like to create an array holding the ID and data when displaying a combo box relating to a specific record. This will remove any problems caused by things with the same name and speed up the system as when you select the item in the combo box the ID is already known instead of having to search through the whole file to find it. Eg on the Setting/Help form when deleting a menu, the orders form when adding a meal from a combo box and the same on the meals forms.

# User Manual see other document