Computing Coursework

Toby Kerslake

January 26, 2016

Contents

Chapter 1

Analysis

1.1 Introduction

Client: Neil Green Job Description: Owns and manages an Independently run gym Contact Information: Sweat Gymnasium LLP 107a Clay Street Soham Ely Cambridgeshire CB7 5HL

Telephone: 01353 721864 Email: info@sweatgym.co.uk

1.1.1 Client Identification

My client is an independent gym owner who is struggling to keep track of his members payment details and use of the facilities. He is also my brother in law. He approached me about creating a system for him after I mentioned I needed a client for coursework and he mentioned that he is having issues with clients not paying him since he doesn't have an accurate, reliable system for keeping track of payments. He also has to manage staff members throughout the day and has to manage his business' accounts, which relies a lot on his current system for keeping track of his members payments. His gym has over a 1000 members which he keeps track of using excel spreadsheets. He made these spreadsheets himself so he has a reputable amount of I.T skills (I go more in depth regarding his and his employees I.T abilities in the Constraints section). I have accumulated the information for the following sections via an interview (presented in full in the appendix) and text messages/email and general conversation as I see him and several of his employees quite frequently but don't always have the chance to document the interaction.

1.1.2 Define the current system

His current system involves using a series of excel spreadsheets to document his clients payment, membership details, personal details and exercise plans. He does this manually by creating new spreadsheets and entering the clients data by hand. His clients give him this data by filling out a membership form and membership agreement and via discussion and another entry form if they need an exercise program made for them. My client uses one spreadsheet (presented in section: Investigation) to keep a record of all the members of the gym including details like their member ID, their contact information and membership details, and another for their payment information (presented in section: Investigation) as well as whether their up to date with these payments. Each member then has their own table which is used to keep track of any exercise routines they may have asked to have planned for them. When the applying member signs up they get given information regarding the gym including payments details so that the can pay via direct debit. Every month the owner checks his mandates to see who has and hasn't made their payment. When a member of the gym leaves all of their information is kept in case they return to their membership in the future. All of the exercise regimes are dated so that it can be referred back to if the member decides they want to change their regime. He then scans these sheets and saves them digitally and uses another physical sheet to document the new regime. A lot of the time he will immediately scan the sheet as members normally request a copy, plus he finds it useful to have a backup especially when dealing with physical forms. All of these spreadsheets and regime forms are kept on the owners workstation computer at the reception desk. Usually he will send the not print the spreadsheets unless he outsources his accounts to another employee or an accountant and he isn't comfortable supplying them with a full digital, or they just find it easier to deal with a physical version. He doesn't keep any of his clients debit/credit card information as all transactions are performed by the members through either direct debit, paying by card at the gym or cash.

1.1.3 Describe the problems

This system imposes some limitations. First of all, all the processes have to be dealt with manually meaing that all the information has to be entered into excel spreadsheets and not in an easier manner like through a digital form which would organise the data efficiently and atomatically instead of the staff having to do it themselves. These spreadsheets are used to log any payments made by a client, whether its cash/card or by direct debit, as well as all the client information that needs to be stored. A password might need to be included in the new system as the current system also isn't secure as any piece of information can be changed at any time by anyone who has managed to access the staff workstation and to be sure of everything debit mandates would have to be looked over, but even then some clients may have payed by cash and thus wouldn't be recoverable. Their

is also the possibility of a computer error that could delete all the documents as they are not backed up anywhere. His gym also now accommodates over 1000 members meaning that he has a large number of spreadsheets for all his clients as well as having his main spreadsheet being exceptionally large meaning the program can lag, especially since the program isn't running on the greatest available hardware. Another issue is that it can be difficult to sort through and search the spreadsheets to find specific information regarding clients, for example if he needs to find they're contact information, excel doesn't have a tool to easily search for that. He also has no easy way to create invoices or just clean organised print outs of any information he needs.

1.1.4 Section appendix

Q. What is your current system?

Our current system involves entering data regarding our clients into excel spreadsheets to keep track of the data we need

Q. And what data is it that you record?

We use the first spreadsheet to record our clients names, contact information, and what membership they have, as well as another for payment details. Each member then has their own table for us to detail any exercise plans that they request for us to create.

Q. What are the limitations of this? This system means that everything has to be done manually and I won't know if Iv'e forgotten to enter whether someones paid or not as their is no way for me to know if I forgot to change part of the spreadsheet or if my client is trying not to pay. It also isn't very secure as anyone could edit it easily. The process can also be very time consuming.

Q. Is their any additional data you would like to be recorded in the new system? I would like to be able to store all of the clients information in one place instead of having to use multiple spreadsheets so I can organise all my data more easily. Though i'm pretty sure I don't need any additional information.

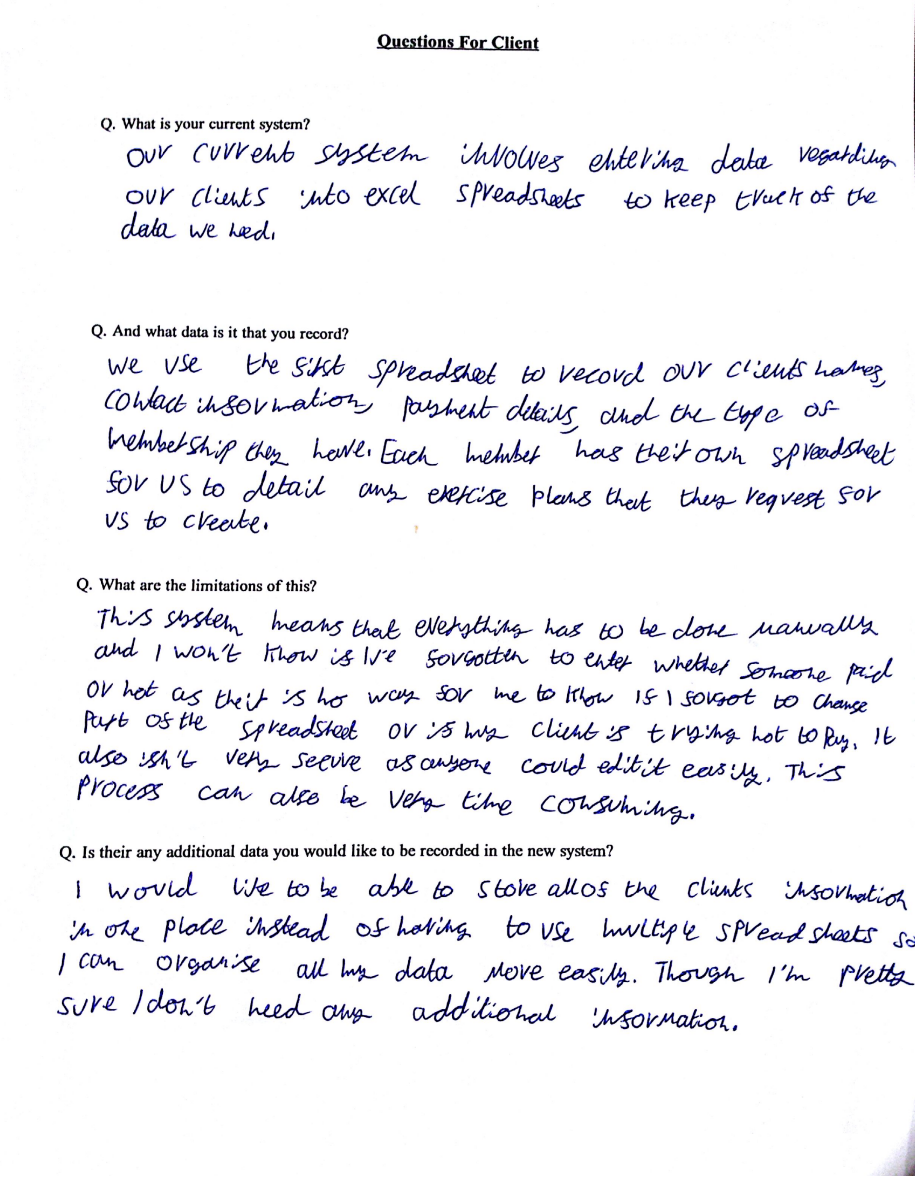
Q. What processes do you normally go through with your current system and how do you implement it? To use my current system I ask my clients to fill out two forms (presented in the investigation section) and then take the information they filled out and enter them digitally into the appropriate excel spreadsheets.

Q.How would you like to operate the new system? I would like to be able to operate the system similar to how I operate my current one with me being able to enter information from forms my clients filled out, or perhaps even with digital forms I can email them to fill out.

Q.What Inputs and outputs would your new system need? I would like to be able to print out a clients information easily in an organised fashion and also be able to create print outs of any other information I would like, possibly to create

an invoice or general report for the client. I would also need some physical forms if possible for my less technically inclined members.

Q. What hardware would this need to run on? This would be run off a reasonably mid spec laptop purchased four years ago. All I really know is it has an i5 processor.



Questions For Client

Q. What is your current system?

Our current system involves entering data regarding our clients into excel spreadsheets to keep track of the data we ned.

Q. And what data is it that you record?

We use the first spreadsheet to record our clients names, contact information, payment details and the type of membership they have. Each member has their own spreadsheet for us to detail any exercise plans that they request for us to create.

Q. What are the limitations of this?

This system means that everything has to be done manually and I won't know if I've forgotten to enter whether someone paid or not as there is no way for me to know if I forgot to change part of the spreadsheet or if my client is trying not to pay. It also isn't very secure as anyone could edit it easily. This process can also be very time consuming.

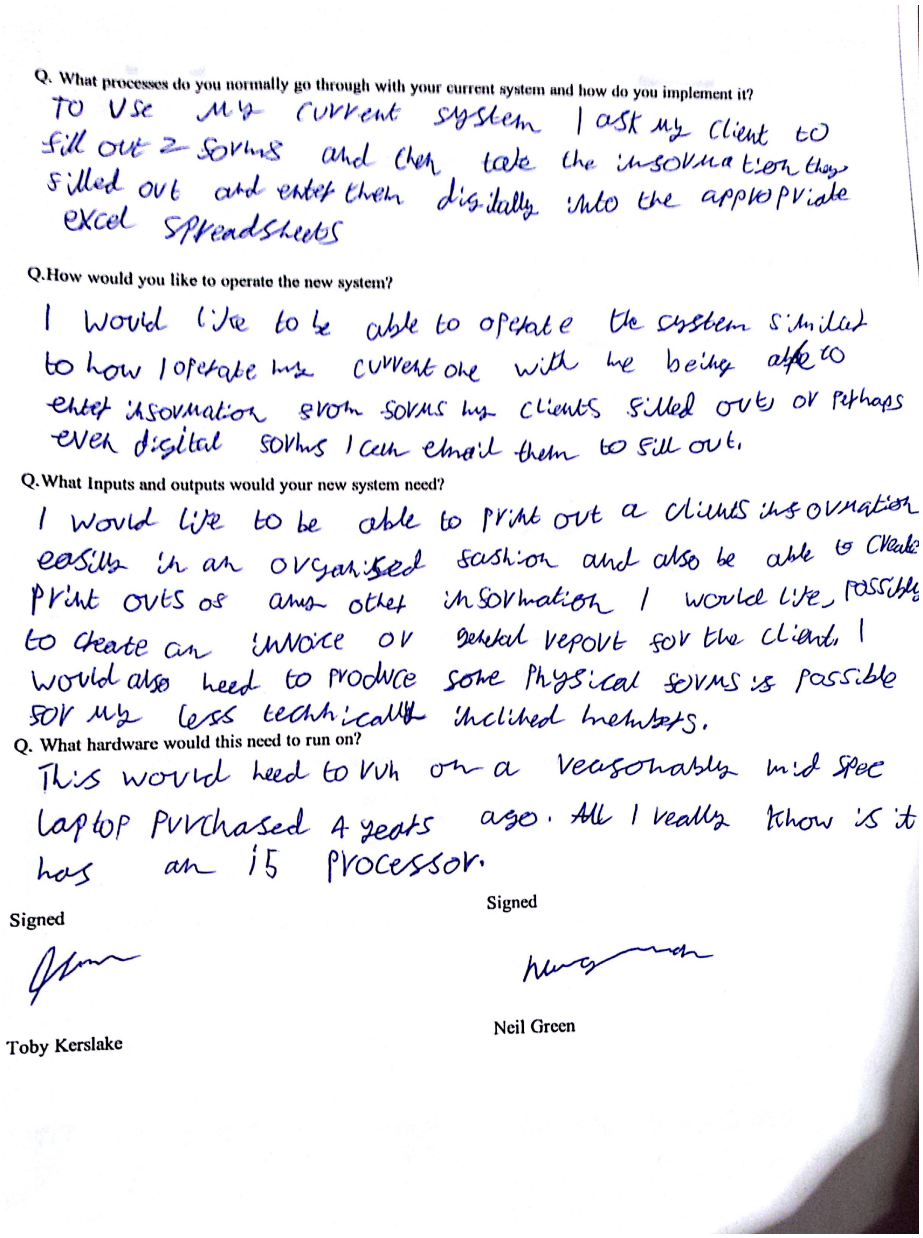
Q. Is their any additional data you would like to be recorded in the new system?

I would like to be able to store all of the clients information in one place instead of having to use multiple spreadsheets so I can organise all my data more easily. Though I'm pretty sure I don't need any additional information.

1.jpg

Figure 1.1: 1st page of the questions aksed to my client

1.2 Investigation



Q. What processes do you normally go through with your current system and how do you implement it?

To use my current system I ask my client to fill out 2 forms and then take the information they filled out and enter them digitally into the appropriate excel spreadsheets

Q.How would you like to operate the new system?

I would like to be able to operate the system similar to how I operate my current one with me being able to enter information from forms my clients filled out or perhaps even digital forms I can email them to fill out.

Q.What Inputs and outputs would your new system need?

I would like to be able to print out a clients information easily in an organised fashion and also be able to create print outs of any other information I would like, possibly to create an invoice or general report for the client. I would also need to produce some physical forms is possible for my less technically inclined members.

Q. What hardware would this need to run on?

This would need to run on a reasonably mid spec laptop purchased 4 years ago. All I really know is it has an i5 processor.

Signed

Toby Kerslake

Signed

Neil Green

2.jpg

Figure 1.2: 2nd page of the questions aksed to my client

1.3 Investigation

1.3.1 The current system

The current system is made of completely manual functions only using computers for data entry and storing the spreadsheets in which the data is entered. I do my best to explain this system below through the use of hierarchy charts, flowcharts, data flow diagrams and the all the physical and digital forms used in the process.

Data sources and destinations

The table below presents all the data curently used in the system, where its come from, where its being stored and gives an example of said data.

Source	Data	Example Data	Destination
Applying Member	Name, Address, Telephone Number, Membership Type, Registration Date, Payment Type, Join Date, Initial Payment Method, How Much	Toby Kerslake, 123 Fakestreet, 01353 666677, Couples, 5/11/2014, Direct Debit, 10/11/2014,	Member Spreadsheet
Applying Member	Initial Payment Type, Registration Date, How Much	Cash, 5/11/2014, £30	Payment Spreadsheet
Member of Staff	Member Number,	100301	Payment Spreadsheet
Member of Staff	A new month is added to the spreadsheet every month and the member of staff enters whether it's been paid for by each member	June '12 Paid	Payment Spreadsheet
Member Interviewed by Staff (and filling out a form)	Exercise Regime	10 Pushups 15kg weights - 20 minutes	Member Programme Card

Figure 1.3: Data Sources and Destinations for the current system

Algorithms

Structure Tables

Gym Sign Up Process

```
1. Information is attained from the client
        1.1 My client gives a form to their client to
            be promptly filled out
        1.2 The client then hands the sheet back into
            reception
        1.3 The information provided on this sheets
            then enter manually into an excel
            spreadsheet
        1.4 Do they have an exercise plan? Then the
            plan is created based on an interview
            with the client while they fill out a form
2.Create Membership Card
        2.1 Make sure all necessary details are
            correct
        2.2 print Membership card
```

Payment Process

```
1. Client enters
2. Are they paying upfront?
2.1. Process their payment
2.2. Enter the appropriate information into the
   correct spreadsheet
2.3 Allow them into the facility
3. Have they payed by direct debit?
3.1. Check to see if the payment went through
3.2. If it did allow them into the facility
3.3 If the payment didnt go through then ask them to
   pay upfront and return to section 2
3.4 If they refuse to pay ask them to leave
```

Pseudocode

Gym Sign Up Process

```

START

Function GetInfo:
        Client fills out form
        Hands form into gym owner
        Owner enters the information into an excel
            spreadsheet
        IF they have an exercise plan:
```

```
                Owner creates a plan based on an
                    interview with client and new form
                    and enters this into  i t s
own spreadsheet
        Owner creates Membership Card
        Owner checks to make sure all the info is
            correct
        Owner prints membership card and gives it to
            the new member

GetInfo

END
```

Payment Process

```
Start

Function ClientPayment:
        Client enters
        IF paying upfront:
                PayingUpFront
        IF paying by direct debit:
                PayingDirectDebit
        IF they refuse to pay:
                AskThemToLeave

Function PayingUpFront:
        Owner processes payment
        Process their payment
        Enter payment information into the
            appropriate spreadsheet
        Allow them into the facility

Function PayingDirectDebit:
        Check to see if the payment went through
        IF payment  d i d n t  go through:
                IF they want to pay up front:
                        PayingUpFront
                ELSE IF they refuse to pay:
                        AskThemToLeave
        ELSE IF the payment did go through:
                Allow them into the facility

Function AskThemToLeave:
        Ask them to leave
```

```
30
31 ClientPayment
32
33 END
```

Flow Charts

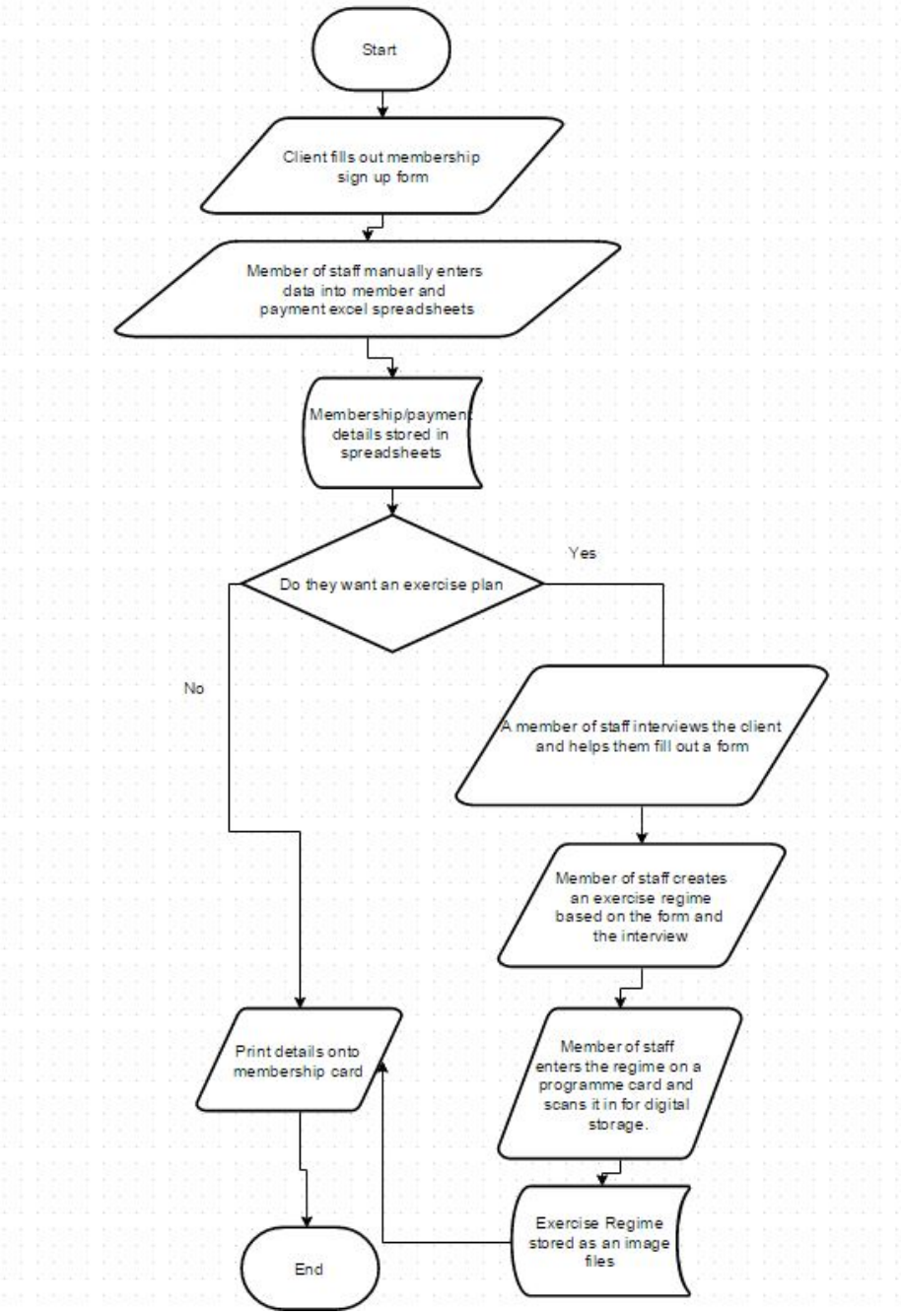


Figure 1.4: Algorithm for the gym sign up process. The form details the process of an applying member giving a staff member their details, the staff member entering them into the spreadsheets and then going through all the extra processes that are required if the applying member wants an exercise regime.

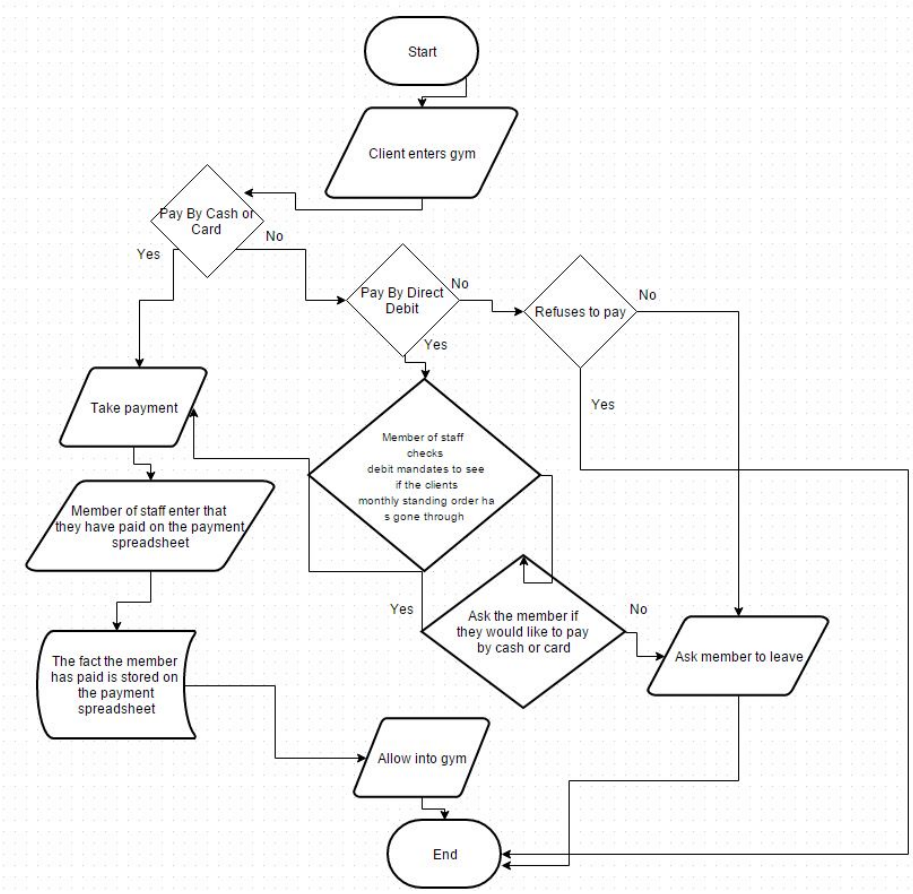


Figure 1.5: Algorithm for the payment process. This details the process the member and staff members go through each month when the member pays either through cash/card or a monthly standing order, and how they deal with people who refuse to pay.

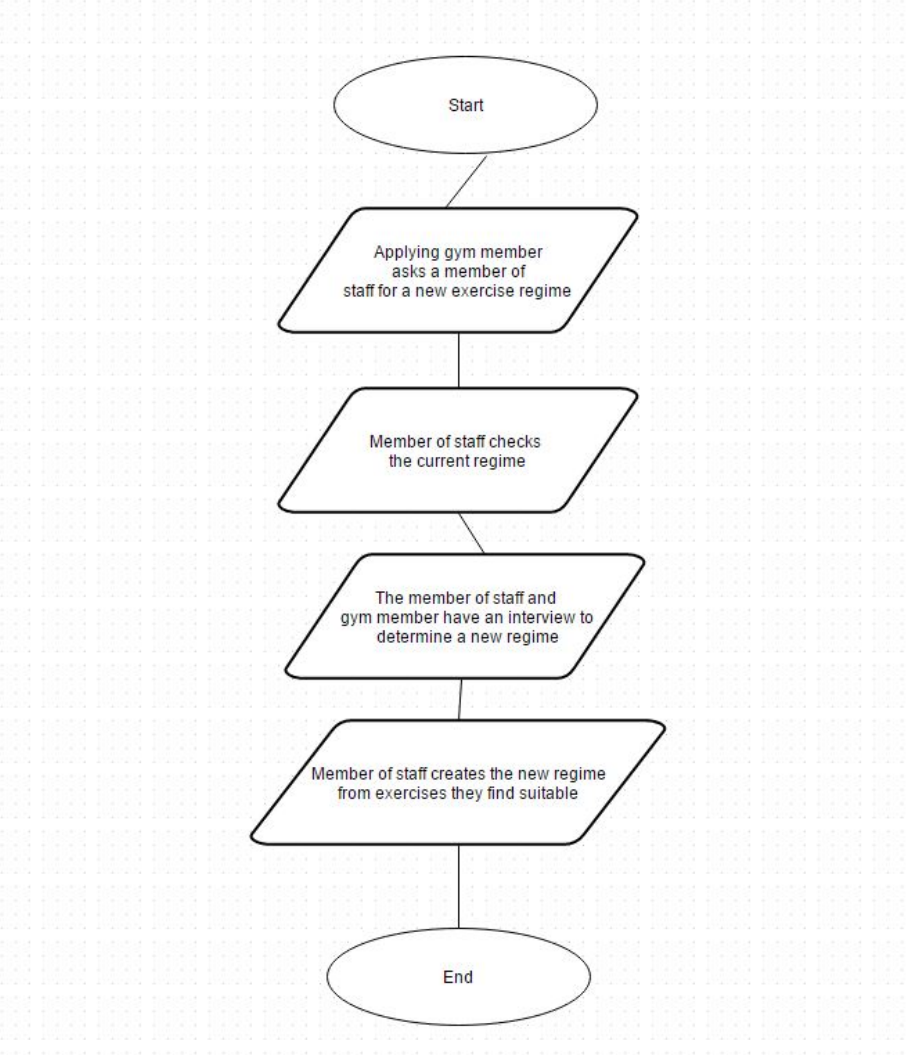


Figure 1.6: Algorithm for the process of creating a new exercise regime. This details how the gym member interviews with a staff member to create a new regime based on exercises they deem appropriate.

Data flow diagram

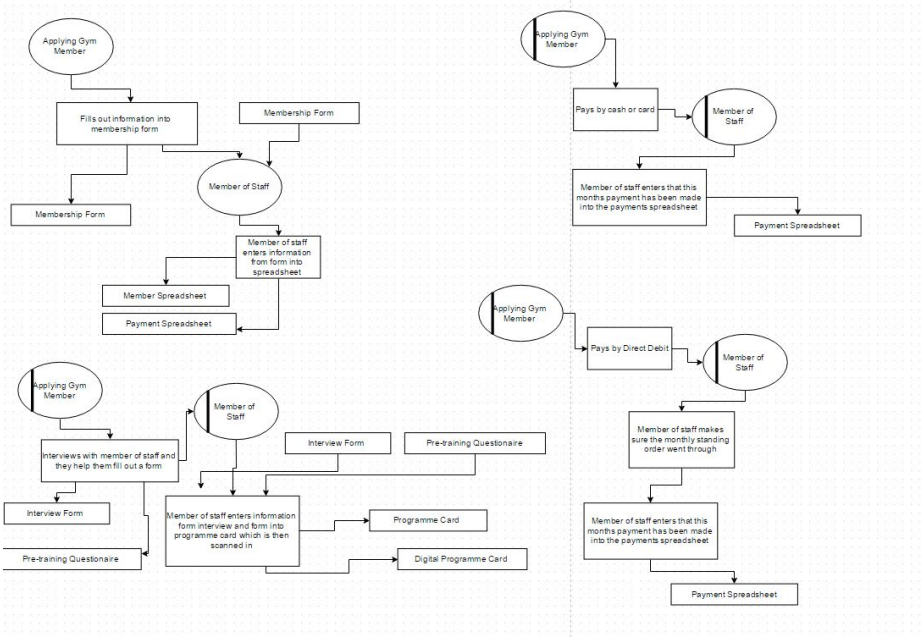
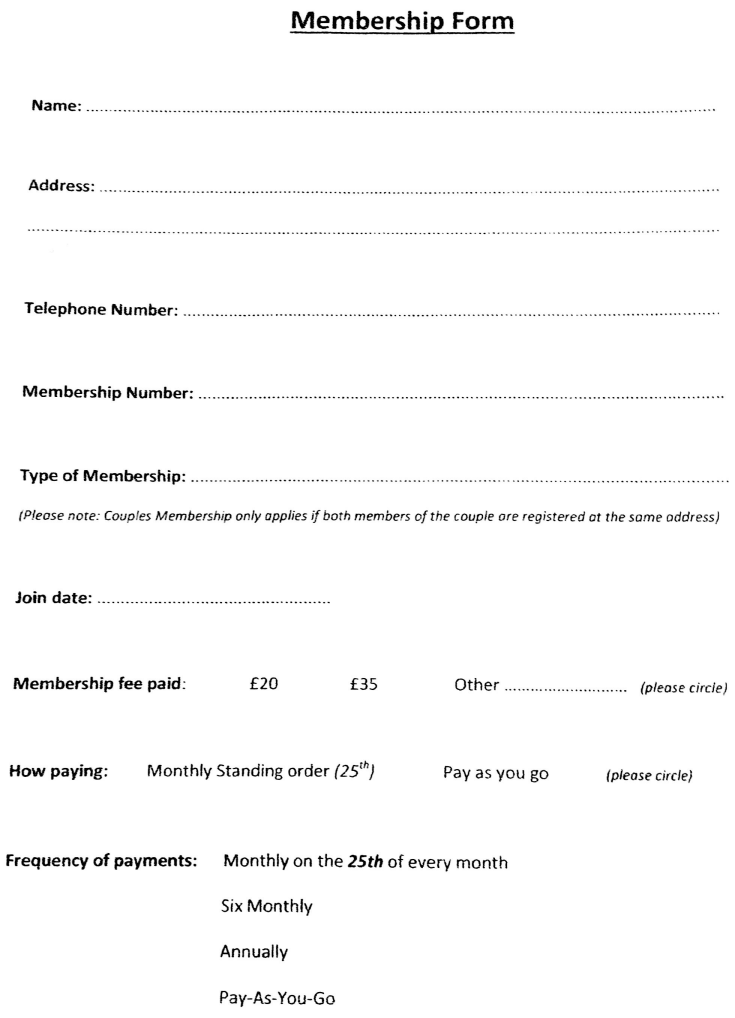


Figure 1.7: Data Flow Diagram For The Current System. These diagrams show where all the data in the current system comes from, where it goes and the process it goes through to get there.

Input Forms, Output Forms, Report Formats



Membership Form

Name: .....

Address: .....

.....

Telephone Number: .....

Membership Number: .....

Type of Membership: .....

(Please note: Couples Membership only applies if both members of the couple are registered at the same address)

Join date: .....

Membership fee paid: £20 £35 Other ..... *(please circle)*

How paying: Monthly Standing order *(25th)* Pay as you go *(please circle)*

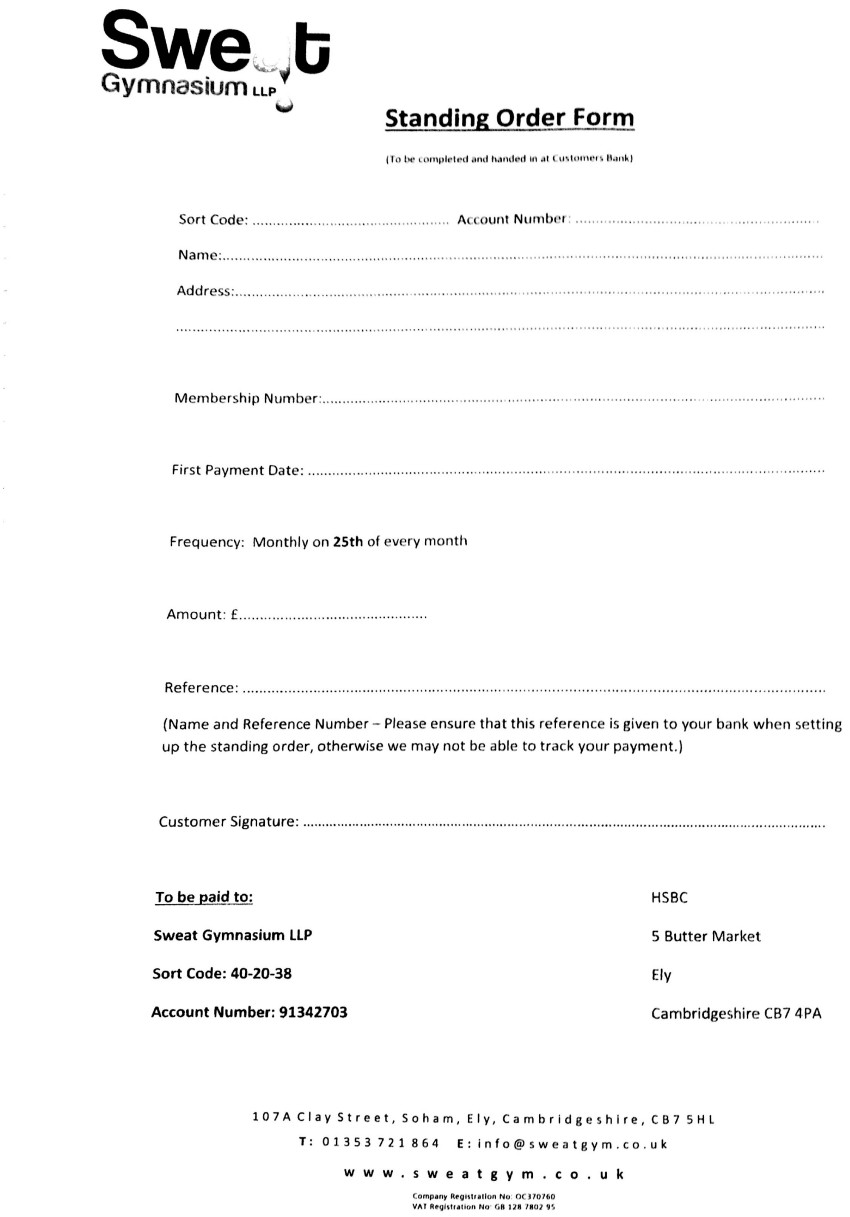
Frequency of payments: Monthly on the ***25th*** of every month

Six Monthly

Annually

Pay-As-You-Go

Figure 1.8: This is the Membership Sign up form. This is printed and handed to applying members upon sign up and can either be filled out at the premises or filled out elsewhere and handed back in to reception. It is used to gather basic information for the membership and payment details spreadsheets.



Sweat Gymnasium LLP

Standing Order Form

(To be completed and handed in at Customers Bank)

Sort Code: ..... Account Number: .....

Name: .....

Address: .....

.....

Membership Number: .....

First Payment Date: .....

Frequency: Monthly on **25th** of every month

Amount: £.....

Reference: .....

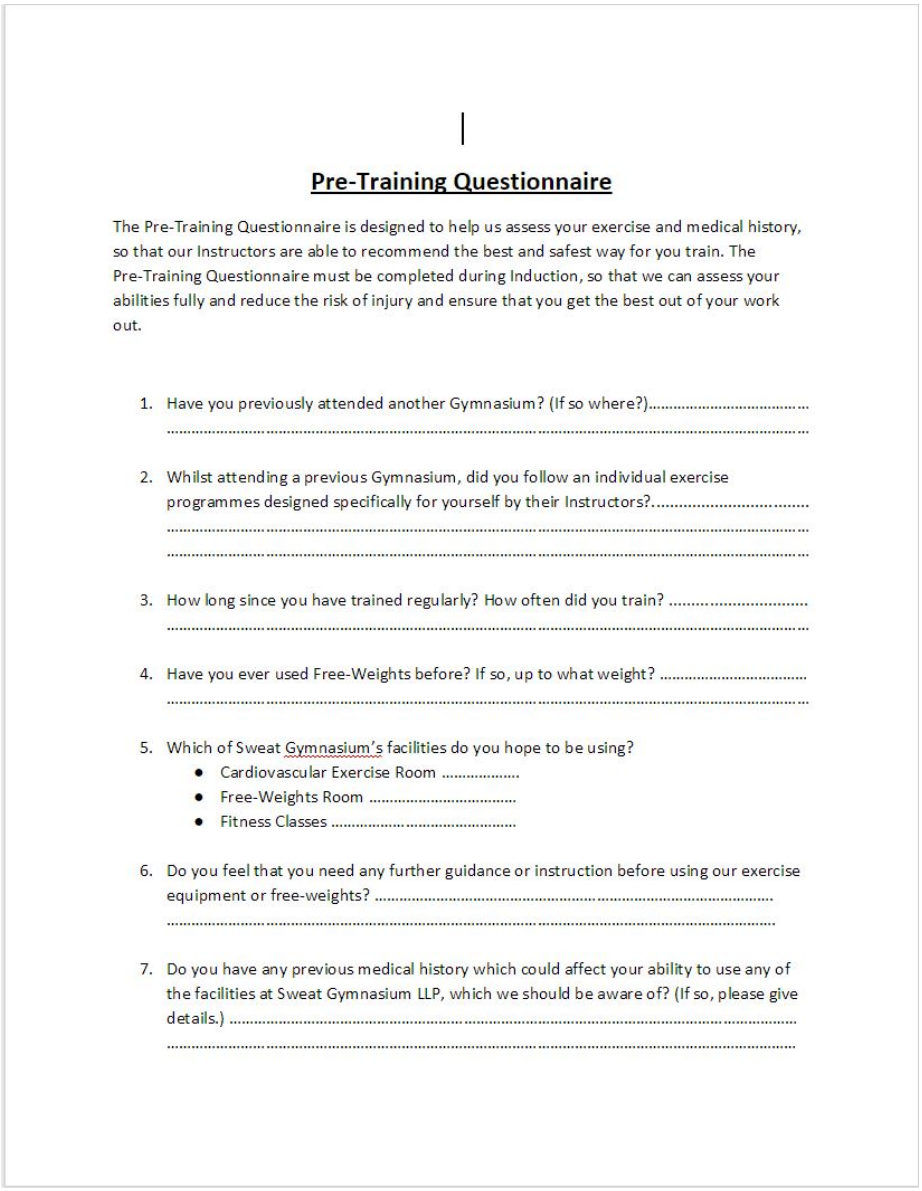
(Name and Reference Number – Please ensure that this reference is given to your bank when setting up the standing order, otherwise we may not be able to track your payment.)

Customer Signature: .....

To be paid to:	HSBC
Sweat Gymnasium LLP	5 Butter Market
Sort Code: 40-20-38	Ely
Account Number: 91342703	Cambridgeshire CB7 4PA

107A Clay Street, Soham, Ely, Cambridgeshire, CB7 5HL
T: 01353 721 864 E: info@sweatgym.co.uk
www.sweatgym.co.uk
Company Registration No: OC370760
VAT Registration No: GB 128 7802 95

Figure 1.9: This is the Standing Order Form. This is printed and handed to the user along with the membership form and is for the member to fill in and hand to their bank so they make the correct payment. Some of this form can be filled in with assistance from a member of staff. For example their membership number.



Pre-Training Questionnaire

The Pre-Training Questionnaire is designed to help us assess your exercise and medical history, so that our Instructors are able to recommend the best and safest way for you train. The Pre-Training Questionnaire must be completed during Induction, so that we can assess your abilities fully and reduce the risk of injury and ensure that you get the best out of your work out.

1. Have you previously attended another Gymnasium? (If so where?).....
   .....
2. Whilst attending a previous Gymnasium, did you follow an individual exercise programmes designed specifically for yourself by their Instructors?.....
   .....
   .....
3. How long since you have trained regularly? How often did you train? .....
   .....
4. Have you ever used Free-Weights before? If so, up to what weight? .....
   .....
5. Which of Sweat Gymnasium's facilities do you hope to be using?
   - Cardiovascular Exercise Room .....
   - Free-Weights Room .....
   - Fitness Classes .....
6. Do you feel that you need any further guidance or instruction before using our exercise equipment or free-weights? .....
   .....
7. Do you have any previous medical history which could affect your ability to use any of the facilities at Sweat Gymnasium LLP, which we should be aware of? (If so, please give details.) .....
   .....

Figure 1.10: Regime Interveiw Part 1

.....

I agree that the information above which I have provided is accurate to the best of my knowledge. I am fully aware that I may be required to supply further information regarding any of the above or my medical history, before being able to use the facilities of Sweat Gymnasium LLP. I understand that refusal to follow the Instructor's guidance when using the facilities at Sweat Gymnasium LLP can result in the inability to train at Sweat Gymnasium LLP.

Printed Name: .....

Signature: .....

Date: .....

Instructors' recommendation: (please circle)

Full Induction Refresher Induction Personalised Fitness Programme

GP Referral No Further Instruction Necessary

Instructors Additional Notes: .....

.....

.....

Instructors Signature: ..... Date: .....

Figure 1.11: Regime Interveiw Part 2. The regime interview/Pre-Training Questionaire is filled out by the applying gym member and a member of staff to determine what the members regime should appropriately consist of. This information is compiled into different exercises and routines stored on the members programme card shown in the next figure. Afterwards these sheets are disposed of.

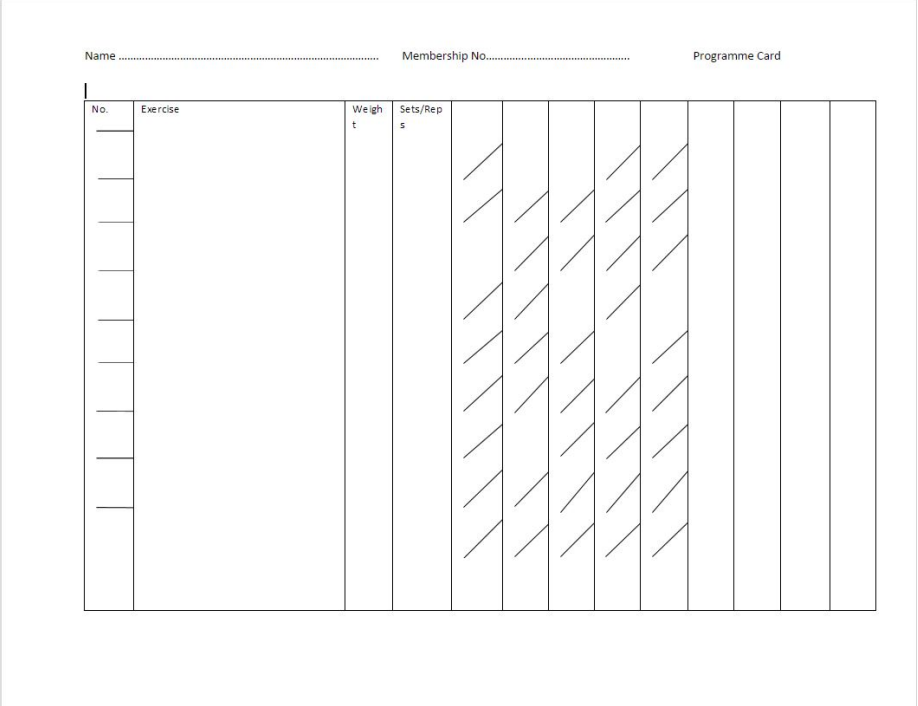


Figure 1.12: The Programme Card is where a Gym members personal regime is written down and organised, with 2 columns to determine the number of reps/weights involved with the exercise. The other columns that contain slashes have no purpose and are just there to fill space. This shows the forms poor design and thus it will probably have to be redisigned for use with the new system.

Membership No.	Surname	First Name		Address	Tel No.
100001	Fakeman	Tobias	P	4 Fake Street, Soham, Ely	01353 624031/ 07846200759
100002	Notreal	George	C	1 Fake Road, Soham, Ely, Cambs CB7 5EP	7990584355
100003	Notreal	Amy	C	1 Fake Road, Soham, Ely, Cambs CB7 5EP	7990584355

Figure 1.13: Member Details Spreadsheet Part 1

Type of Membership	How Paid	Amount	S.O or Cash	When Joined
Peak - 6 Months Upfront	£160 cash upfront (01.02.13-01.08.13)	£160	Cash upfront	23.06.12
Couples Peak (100003 - A. Notreal)	£47 paid for August	£47	S.O.	23.06.12
Couples Peak (100002 - G. Notreal)	£0 - see above	£0 - see a	S.O.	23.06.12

Figure 1.14: Member Details Spreadsheet Part 2

Registration Fee Paid	Date of Induction	Comments
£20 cash paid 23.06.12	25.06.12	6 months peak upfront (01.02.13 - 01.08.13)
£35 cash paid 23.06.12	25.06.12	Changed to Couples Peak (see 100003 - J. Fletcher) from August 2013
£0.00 23.06.12	Hattie: 28.06.12	Changed to Couples Peak (see 100002 - A. Fletcher) from August 2013

Figure 1.15: Member Details Spreadsheet Part 3. All the information stored in this spreadsheet was gathered from previous forms or noted down by a member of staff during the members induction and then entered into the spreadsheet. This is stored locally on a workstation in the gym where only authorised users can gain access to it. The spreadsheets are rarely printed out except in special circumstances like accounting purposes.

Membership No.	Full Name	Type of Membership	How Paid	Amount	Registration Fee (Date)
100001	Tobias Fakeman	Peak - 6 Months Upfront	£160 cash upfront (July-Dec)	£160	£20 paid 23.06.12
100002	George Notreal	Couples Peak (see 100003 - A. Notreal)	£47 paid for August	£47	£35 paid 23.06.12
100003	Amy Notreal	Couples Peak (see 100002 - G. Notreal)	£0 paid for August - see above	£0	£0 - see above

Figure 1.16: Payment Details Spreadsheet part 1

How Pay?	June '12	July '12	Aug '12	Sept '12	Oct '12	Nov '12	Dec '12	Jan '13	Feb '13	Mar '13	Apr '13	May '13	June '13
cash upfront	x	Paid	Paid	Paid	Paid	Paid	Paid		Paid	Paid	Paid	Paid	Paid
S.O.	x	Paid	Paid	Paid	Paid	Paid	Paid	Paid	Paid	Paid	Paid	Paid	Paid
S.O.	x	Paid	Paid	Paid	Paid	x	x	x	x	x	x	x	x

Figure 1.17: Payment Details Spreadsheet part 2. This spreadsheet contains payment details collected from the previous forms and during the induction period. It also keeps track of every month and whether the member has paid for said month.

1.3.2 The proposed system

My proposed system uses a database to keep track of all the data that my client previously kept in an excel spreadsheet using a program created in the python programming language using the PyQt 4 library. This is because its the language I'm most comfortable with and allows for the creation of intuitive UI and the creation of an executable program using the software cx freeze ensuring that the system will be easily usable by my client.

Data sources and destinations

Sources and Definitions - Proposed 2.JPG

What is it	Source	Process	Example	Destination
Name	Member	Member enters his name into digital form/ writes it on physical form and is then entered into the database either manually or automatically via the form	Toby Kerslake	Members Table in Database
Address	Member	Member enters his address	123 fake lane	Members Table in Database
Tel. Num	Member	Member enters his number	01353 778934	Members Table in Database
Type of Membership	Member	Member enters his desired membership type	Couples	Members Table in Database
Join Date	Member	Member enters date in which he desires to join	12/05/2015	Members Table in Database
Date of Induction	Member	Member enters the date of their induction	15/05/2015	Members Table in Database
How Paid	Member	Member enters how they have payed for their membership	Card	Members Table in Database

Figure 1.18: Data Sources and Destinations

Sources and Definitions - Proposed.JPG

Amount	Member	How much the member paid for registration	£35	Members Database
Registration Fee	Owner	How much the members registration cost	£35	Members Database
Registration Date	Member	The date of registration	12/05/2015	Members Database
Payment Type	Member	Member enters how they are paying	Pay as you go	Members Database
Date Of Payment	Member of Staff	The date which payment is due	12/06/2015	Payment Table in
How Much?	Member Of Staff	How Much they have to pay	£20	Payment Table in
Paid	Member of Staff	Whether the client has paid for a given month	Yes	Payment Table in
Exercise Name	Member/Owner	The name of an exercise	10 Star Jumps	Exercise Database
Exercise Description	Member of staff/owner	Descriptioin of the exercise	Jumping up and Down	Exercise Database

Figure 1.19: Data Sources and Destinations 2

Proposed Sources 3.JPG

Specific Description	Member of staff/owner	More specific description of the exercise more tailored for the members needs	10 star jumps to increase indurance	Regime Table in Database
Start Date	Member of staff/owner	The date the member started this regime	5/11/14	Regime Table In Database
End Date	Member of staff/owner	The date the member finished doing this regime	17/01/15	Regime Table In Database

Figure 1.20: Data Sources and Destinations 3

Data flow diagram

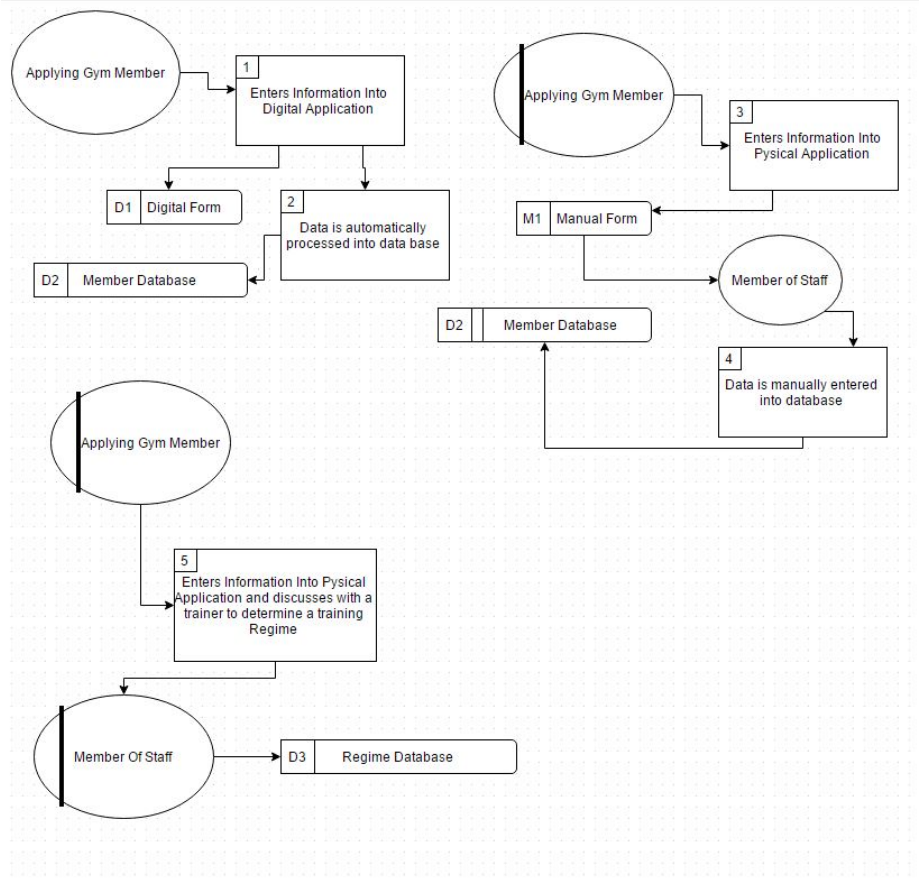


Figure 1.21: Data Flow Diagram

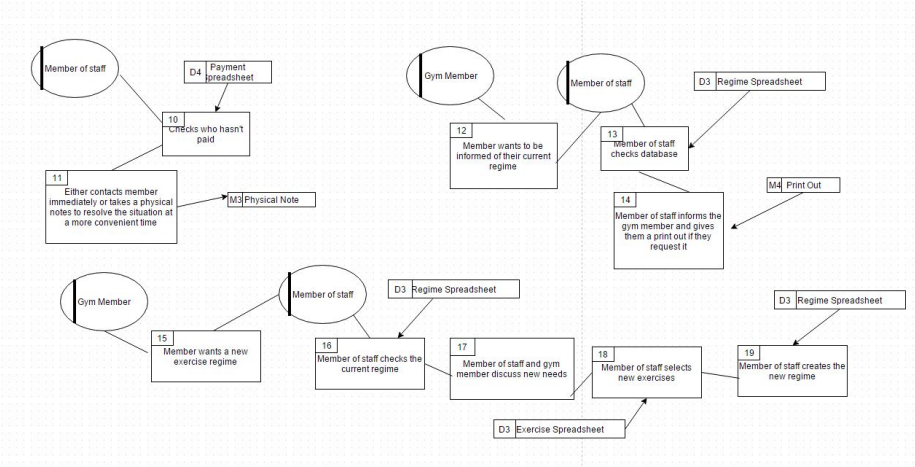


Figure 1.22: Data Flow Diagram

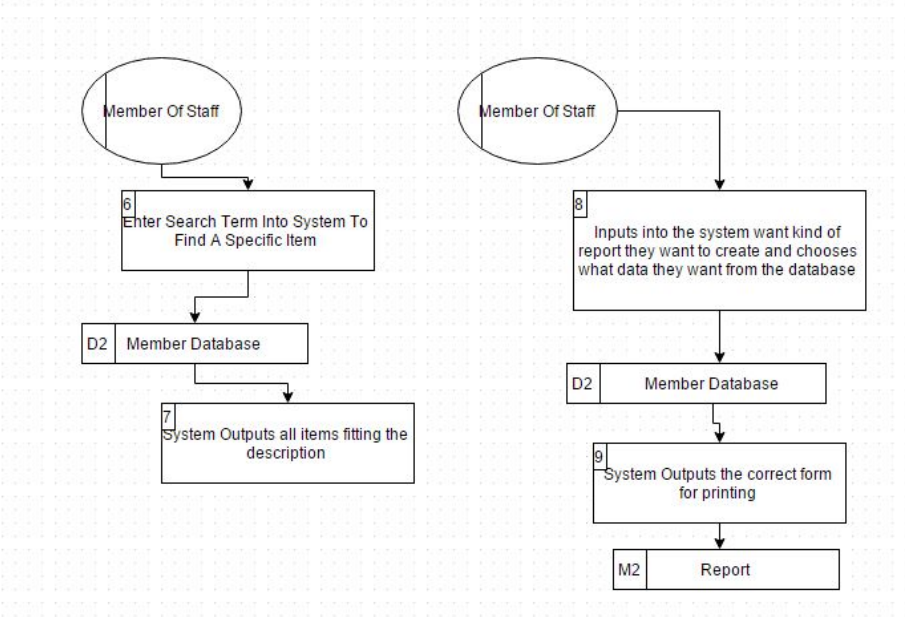


Figure 1.23: Data Flow Diagram. These diagrams show where all the data in the proposed system comes from, where it goes and the process it goes through to get there.

Data dictionary

Entity	Attribute	Data Type	Length	Constrain	Description
Member	Member No	Int	(10)	Primary Key	The unique membership ID
Member	Name	String	(30)	Not Null	The name of the member
Member	Address	String	(30)	Not Null	The members address
Member	Telephone Number	String	(10)	Not Null	The members preferred contact number
Member	Type of Membership	String	(10)	Not Null	The type of membership
Member	Join Date	Datetime	(8)	Not Null	The date the member joined
Member	Date Of Induction	DateTime	(8)	Not Null	The date of the members induction
Member	How Paid	String	(10)	Not Null	How the member payed for their membership
Member	Amount	int	(2)	Not Null	How much the member payed for

Figure 1.24: Data Dictionary

					their membership
Member	Registration Fee	int	(2)	Not Null	How much does the member have to pay for registration
Member	Registration Date	Date/Time	(10)	Not Null	The date of retgistration
Payment Details	Member No	Int	(10)	Primary Key	The unique membership ID
Payment Details	Payment Type	String	(22)	Not Null	How the client pays
Payment Details	Date Of Payment	Datetime	(8)	Not Null	The Date the payment is due
Payment Details	How Much	Int	(2)	Not Null	How much the client needs to pay
Payment Details	Paid	Boolean	(1)	Not Null	Whether the client has payed
Regime	Member No	Int	(10)	Primary Key	The unique membership ID
Regime	ExerciseID	int	(10)	Not Null	The unique exercise identifier
Exercise	ExerciseID	int	(10)	Primary Key	The unique exercise identifier

Figure 1.25: Data Dictionary 2

Exercise	Name	String	(20)	Not Null	The name of the exercise
Exercise	Description	String	(50)	Not Null	Description of the exercise
Regime	Specific Description	String	(50)	Not Null	More specific description of the exercise tailored for the member
Regime	Start Date	Datetime	(8)	Not Null	The date the member started the regime
Regime	End Date	Datetime	(8)	Not Null	The date the member finished doing the regime

Figure 1.26: Data Dictionary 3

Volumetrics

I have decided on an initial size of 2000 members since the gym is nearing that number of members and will inevitably achieve this amount of clients. My client has said the amount of clients he gains isn't consistent as people often register in different sized groups. Although over the past year he has gained 560 new members and roughly 400 members the year before, and the year before that. This means that we can estimate that in about a year and a half the gym will exceed 2000 members. The size of the database can be increased at a later date when needed.

Each member has 11 - 16 fields of data in total, with each data field taking up 1 KB of data.

16 * 1KB * 2000 = 32000 KB 32000 / 1024 = 31.25 MB

There is also the exercise tables and regime tables which will take up roughly 10,000 KB of data assuming each member will have an average of 10 exercises each in their regime, each exercise taking up 1KB of data. The exercise table would then likely come up to 300 KB of data as currently the owner estimates that he currently offers 300 different variations of exercises.

10,300 / 1024 = 10.05 MB

The actual program should then take up a small amount of space that will probably take up about 3MB.

31.25 MB + 10.05 + 3MB = 44.30 MB

1.4 Objectives

1.4.1 General Objectives

- Clear/easy to understand layout structure for member/client records viewing/observation
- Clear/easy to understand layout structure for all member/client data input
- Clear/easy to understand layout for creating and printing invoices/ member records
- Clear/easy to understand digital/physical input forms for users to enter data
- Secure system thats password protected so that only appropriate staff members can use the system
- Cerain options like deleting items from the database restricted to only an administrator/ sepearte password that only higher level staff members can access
- Presentable, minimalistic GUI that can be easily understood and interacted with by the users
- Appropriate Screen outputs for example dialog boxes and the presentation of the database presented in the GUI mockup.

1.4.2 Specific Objectives

Client/Member Record Viewing/Data Input

- Easy to use Simple Gui with clearly labelled buttons to choose an option (as shown in the mock-ups I presented my client).
- Their should be buttons for opening a table, add to it, editing it, deleting an item, creating an invoice or physical record, and searching for something specific.

- Each of these buttons should open up either drop down menus or entirely new windows, depending on which is more user friendly, and display further options within those commands.
- The interface for entering new information should be easy to use and clearly labelled for adding each attribute to the specified table.
- Their should be an easy/ifficient way to switch and search between all of the tables in the database.
- Avoid more novice users accidentally deleteing items by having a secondary password for higher level access.
- Store backups of the database online using a system like dropbox or GitHub so that in the event of data corruption there would be recent version to backup to
- Storing the latest version of the spreasheet locally also adds an extra layer of security as the most recent version wont be easily accessable exept by people with direct access to the workstation.

System Outputs

- Easy to create reports containing all of a specific clients information in an organised manner on as little paper as legibly possible.
- Easy to create invoices for members based off of the information stored in the tables.
- Reports should include Invoices, database table printouts, member reports giving details selected from the databases.
- These reports should be printable so that they can be given to members and stored physically

Input Forms

- Easy to fill out digital input forms for entering new data into the database tables instead of a more complicated sql query based system.
- Printable versions off these forms that can be created to be filled out by hand by a client that can then have their information manually entered into the system by a member of staff.

1.4.3 Core Objectives

- Easy to use Simple Gui with clearly labelled buttons to choose an option (as shown in the mock-ups I presented my client).
- Their should be buttons for opening a table, add to it, editing it, deleting an item, creating an invoice or physical record, and searching for something specific.

- Each of these buttons should open up either drop down menus or entirely new windows, depending on which is more user friendly, and display further options within those commands.
- The interface for entering new information should be easy to use and clearly labelled for adding each attribute to the specified table.
- Their should be an easy/ifficient way to switch and search between all of the tables in the database.
- Avoid more novice users accidentally deleteing items by having a secondary password for higher level access.
- Store backups of the database online using a system like dropbox or GitHub so that in the event of data corruption there would be recent version to backup to
- Storing the latest version of the spreasheet locally also adds an extra layer of security as the most recent version wont be easily accessable exept by people with direct access to the workstation.
- Easy to create reports containing all of a specific clients information in an organised manner on as little paper as legibly possible.
- Easy to create invoices for members based off of the information stored in the tables.
- Reports should include Invoices, database table printouts, member reports giving details selected from the databases.
- These reports should be printable so that they can be given to members and stored physically

1.4.4 Other Objectives

- Easy to fill out digital input forms for entering new data into the database tables instead of a more complicated sql query based system.
- Printable versions off these forms that can be created to be filled out by hand by less technically inclined clients that can then have their information manually entered into the system by a member of staff.

1.5 ER Diagrams and Descriptions

1.5.1 ER Diagram

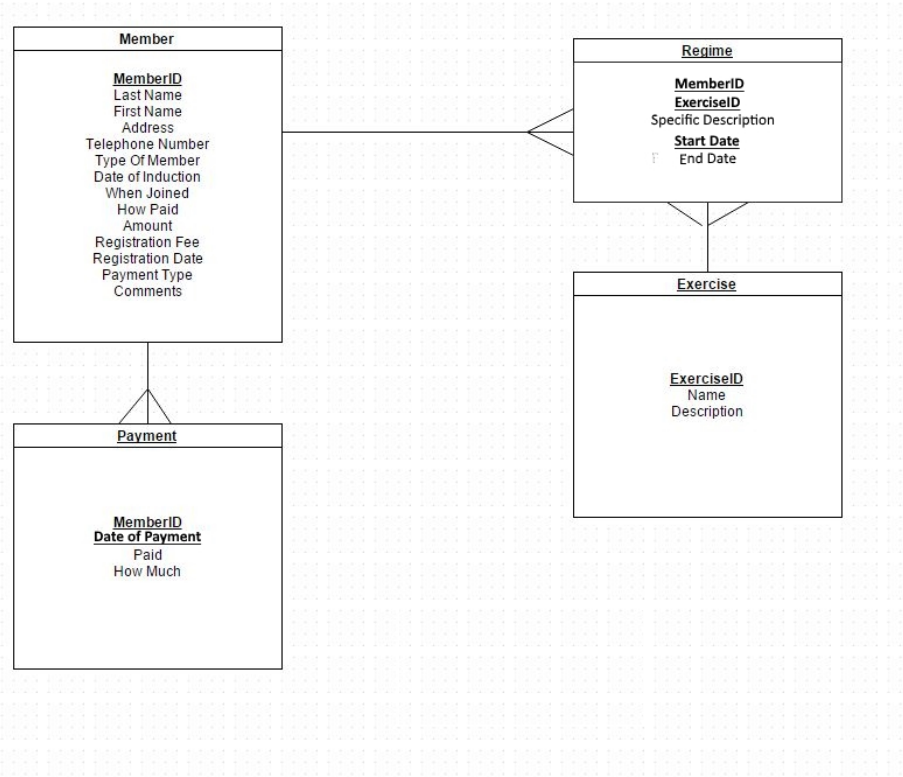


Figure 1.27: ER Diagram

1.5.2 Entity Descriptions

Membership Details(**Membership No.**, Last Name, First Name, Address, Telephone No., Type Of Membership, Date of Induction, When Joined, How Paid,amount, registration fee, Registration Date, Payment Type, Comments)

Payment Details(**Membership No.**,**Date of Payment**, How Much, Paid)

Regime(**Membership No.**,**ExerciseID**,Specific Description, **Start Date**, End Date)

Exercise(**ExerciseID**, Name, Description)

1.6 Object Analysis

1.6.1 Object Listing

Membership Details

Payment Details

Regime

Exercise

1.6.2 Relationship diagrams

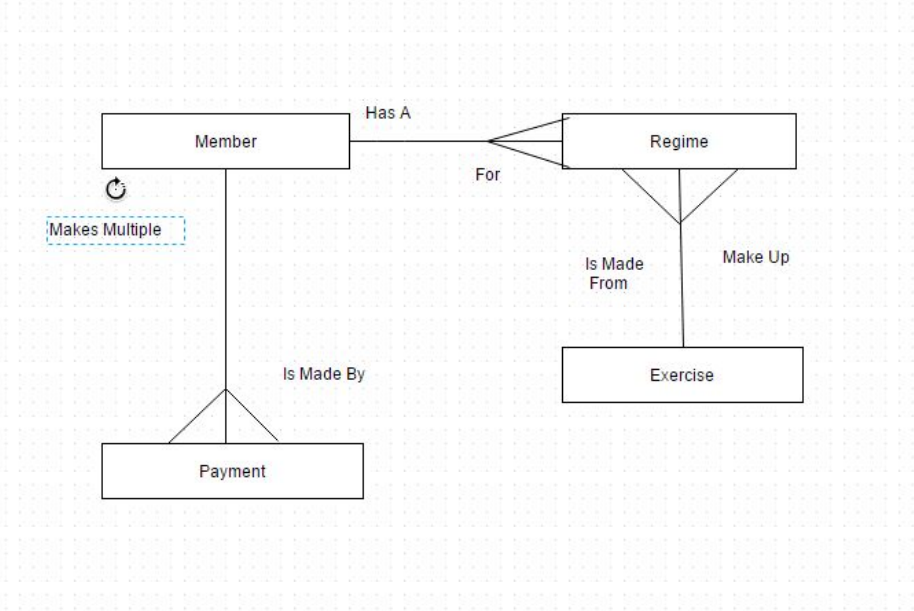


Figure 1.28: Relationship Diagram

1.6.3 Class definitions

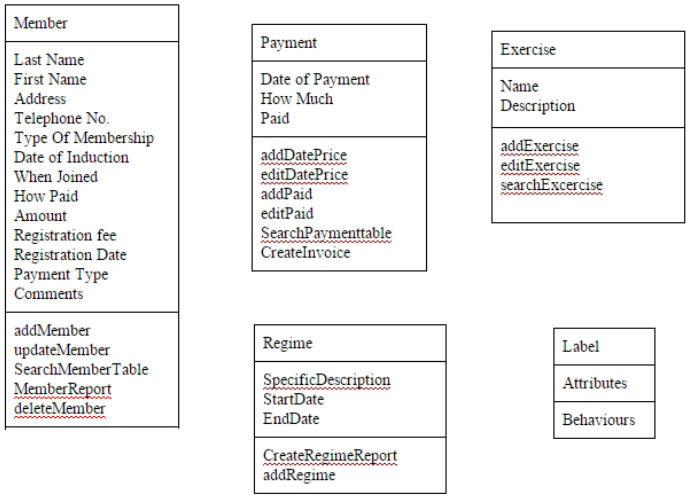


Figure 1.29: Definitions

1.7 Other Abstractions and Graphs

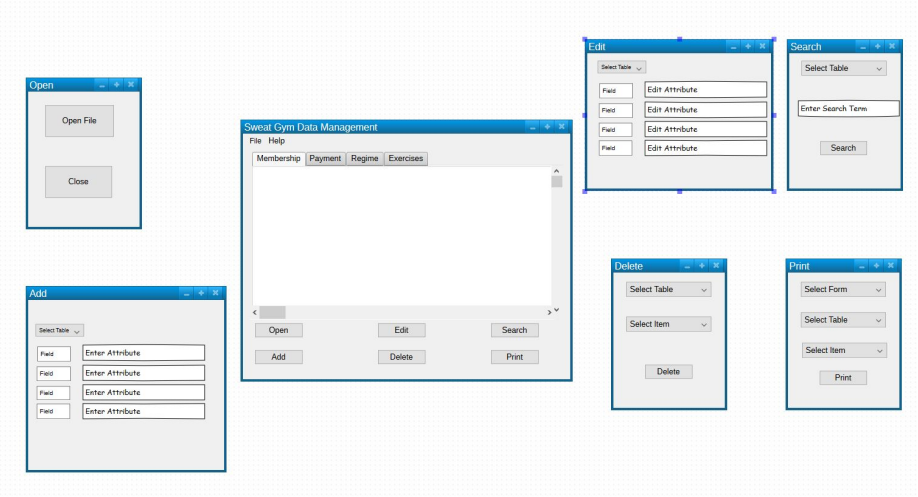


Figure 1.30: This is the early preliminary designs for the GUI for the proposed system. It shows a main window (with a white box representing where the database will be presented) containing 6 push buttons that open into the 6 respective dialog boxes.

1.8 Constraints

1.8.1 Hardware

The owner currently uses a Notebook laptop powered by an intel 1st generation i5 processor and 4GB of RAM. He already uses this laptop to run his current system which uses more resources and power than this system will hopefully require, but if not he still has more than enough horse power to run the new system. His use of a laptop is convienient as it means his system is portable so he doesn't need to do any data entry (if required) confined to his office and away from his client if he doesn't want too and it also means he has a battery so in the case of a power outtage he wont lose any data. Though it is worth noting that he may soon be upgrading to a more powerful desktop soon so while this lowers portability, he has more power for the proposed system and he can easily have a battery backup in the form of a UPS(Uninterptable Power Supply).

All members will be running this program off the same workstation so the databases don't need to be primarily stored online and stored locally as the files don't need to be portable. This also adds a level of security.

The only hardware constraint this laptop proposes is the size and resolution of the screen. As it is only 1440p x 900p and 15 inches diagonally this doesn't give him much room to observe over 1300 clients without excessive amounts of scrolling. Fortunately this could be easily solved by either upgrading to a desktop or simply using an external monitor of a higher resolution.

1.8.2 Software

My client has no preference on what software can or cannot be used as long as its easily accessable to him as he isn't the most computer literate person. Its worth noting that although he has no particular requirements for software he isn't willing to learn a new operating system so it will have to run on windows 7. This is not a problem since the proposed system will be developed on a windows os and should be fully compatable. The client wants the system to be ifficient so the databases are going to be programmed in SQL and the system itself is going to have two levels of password protection so that the data stays secure.

1.8.3 Time

My client has given me no personal deadline for this project so the only one I have is the one set by AQA which at the time of writing is April 2015. Although my client doesn't need the new system immediately he is willing to use it as soon as possible and willing to test it before its finished providing its in a fully functioning state.

1.8.4 User Knowledge

Although the owner and staff members aren't specifically trained in I.T they are all completely able to use a computer when given the necessary instruction and support. Most of the staff only use computers for email and web browsing, and occasioinally writing things with a word processing package, as well as the current system which is computer based.

1.8.5 Access restrictions

Access to the proposed system should only be available to members of sweat gymnasium staff and thus the system may need a password although my client insists that only authorised people will have access to the computer, which is password protected itself so the password may be set to optional incase he wants to deactiate it. Due to the system containing private and confidential data on individuals it will have to comply with the Data Protection Act and appropriate action will be taken to do so though the care of the programs security is mostly

in the hands of the staff and owner. Although the password for the program may be made optional I insisted that the database be password protected so that people without clearance can't open the database in another browser for malicious or freudulant purposes. If a password is used for the program which it likely will be then a secondary password will be used for the delete option so that no novice users can accidentally delete an item as only higher users who are authorised (like the manager) will have access to that password.

1.9 Limitations

1.9.1 Areas which will not be included in computerisation

For the less technically inclined who don't have email or don't feel that digital forms are of convienience there will be physical forms for them to fill out that would then require their data to be entered into the proposed system manually by the owner or a member of staff. The forms that need to be filled out to determine an exercise regime/programme will also stay physical as they have some very specific questions that can't easily be answered with enough detail in a digital form since the applying member may need assistance from a member of staff and consultation from a trainer.

1.9.2 Areas considered for future computerisation

Although the invoices and printout records of each member could be stored digitally this would be a bad idea for a number of reasons. First of all the printouts could need updating at anytime and if they need a new physical record the gym will need an up to date one instead of an older one they saved from a couple of months before. Another reason for this is that Iv'e been told by the owner that he would prefer to keep hard copies of invoices over digital so that he not only has a physical record of the invoice for accountancy reasons and so he can send one to any of his clients, even the ones with little computer knowledge. The forms also change repeatedly so if they were saved an older version could be printed out instead by accident. This will be considered for future computerisation as it may come in handy if the owner decides he may want digital copies of forms kept if he believes hes going to need to repeatedly print off an unchanged form in the future, or needs a digital record of a table in a previous state. So a fucntion to save a form to a text file or binary file that can be reread by the program to be printed off in the future may be added.

1.10 Solutions

1.10.1 Alternative solutions

Improving Spreadsheets

- Advantages
  - More familiar structure to the current system
  - Easier to edit to adapt to new features
  - Easy to make back ups
  - Microsoft support available if anything goes wrong with the software
  - Gets rid of all the data duplication in the current spreadsheets
- Disadvantages
  - Still prone to the same security issues as the old system
  - All data still has to be entered manually
  - No easy and specific way to sort through data
  - No easy to create invoices or printable records

Online HTML Based Application

- Advantages
  - Easy to access using any computer running any operating system and even mobile devices
  - Low system requirements as it runs in browser based on HTML
  - Cloud based so backups can be kept securely on error checking hardware/servers
  - SQL has great web integraion for databases
  - Wouldn't require that anything would need to be installed
- Disadvantages
  - I don't know HTML/CSS and javascript that well and would have to spend large amounts of development time learning the language
  - Online applications have massive sequrity risks and ths could open the sytem up to attacks like SQL injection
  - If the gyms internet connection becomes innadequate for any reason then he system couldn't be accessed
  - Web hosting can be very expensive

Paper Based Manual System

- Advantages
  - Easy for everyone to understand as it requires no technology
  - Can be used immediately as there will be basically no development time
- Disadvantages
  - Requires a large amount of space to store so many physical documents
  - No easy way of making backups except for maybe carbon copies
  - Can easily be damaged and destroyed/lost
  - Has no easy search method and could easily become un-organised
  - No way to edit documents

Command Line Application Programmed in Python implementing a database saved to text or binary files

- Advantages
  - Relatively easy to program with a short development time
  - Easy to use if users are taught appropriately
  - Can be programmed to do anything required and easily modified in the future
  - Saving the databases to one of these formats is relatively easy to programm and manage.
- Disadvantages
  - Commandline can be difficult for some people to understand
  - Executing the program may be difficult for some users
  - Using text files and binary files is very insecure as they can be easily edited, manipulated and read by anyone with moderate I.T knowledge.
  - Using these formats means that all sorting methods and search methods have to be programmed be me instead of using a function from a library like sqlite3 resulting in a considerable use of time and resources that isn't necessary.

Command Line Application Programmed in Python implementing a database programmed and managed in SQL

- Advantages
  - Relatively easy to program with a short development time

  - Easy to use if users are taught appropriately
  - Can be programmed to do anything required and easily modified in the future
  - Uses relational databases that will be easily understandable as it will be designed to my clients specifications when compared to the previous excel spreadsheets as well as being more secure than a text or binary file as well as being easier to manipulate with the sqlite3 library.
  - With the database being saved as a database file, if the program encounters some form of unkown bug down the line the database can still be opened using another SQL browser program
- Disadvantages
  - Commandline can be difficult for some people to understand
  - Executing the program may be difficult for some
  - SQL is slightly more difficult to programm compared to just using text files or binary files
  - SQL still isn't totally secure as using digital input forms means that it may be susceptable to SQL injection and other security faults.

GUI Based Application Programmed in Python using the PyQt Library implementing a database programmed and managed in SQL.

- Advantages
  - Easy to program as I have experience with both python and PyQt so the development time will be quite short
  - Can be programmed to do anything the client requests
  - Easy to use as entirely GUI based so users only need basic computer knowledge
  - That program can be made into an executable using cx freeze so its easy to open and doesnt require the install of any additional programs
  - Can be programmed to do anything required and easily modified in the future
  - Uses relational databases that will be easily understandable as it will be designed to my clients specifications when compared to the previous excel spreadsheets as well as being more secure than a text or binary file as well as being easier to manipulate with the sqlite3 library.

  - With the database being saved as a database file, if the program encounters some form of unkown bug down the line the database can still be opened using another SQL browser program

- Disadvantages
  - Not as easy an adjustment as updating the current spreadsheets would be
  - Not as easy to create as a command line based program
  - SQL is slightly more difficult to programm compared to just using text files or binary files
  - SQL still isn't totally secure as using digital input forms means that it may be susceptable to SQL injection and other security faults.

1.10.2 Justification of chosen solution

My Chosen solution is too use an application with a GUI programmed in python with the library PyQt implementing a database programmed and managed in SQL. This is because of a number of reasons:

- I am familiar with python and PyQt and know how to use them as well as knowing how to use and disect their documentation if I ever get lost
- Python is easy to install and I can easily use it to debug the program on the gyms workstation if needed incase of emergency though the program will be distributed to my client as a executable
- If its more accessable to the client I can even make an executable of the program using a program like cx freeze
- Programming in python gives me the opportunity to present and demonstrate my knowledge of complex programming features like Object Oriented Programming and recursive programming
- Python gives me he flexibility to program the system to my clients exact specification as its a fully featured programming language
- The use of relational databases can be implemented using python and the SQLite3 library meaning it will be easy to store and manipulate the data required using SQL as oppossed to just a text file or a binary file or an Excel spreadsheet
- The use of SQL databases is that it is much more secure than simply saving the data to a text file, or pickling the data to create a binary file, and even more secure than a password protected or encrypted Excel spreadsheet. It also means that in the event that the program encounters an issue that the database can be opened in another SQL Browser, even if the interface

wont be as suited to the clients needs as the one I will have programmed in the chosen solution.

- This also means that the database can be presented in any format/design that is desired as long as its possible to program

Chapter 2

Design

2.1 Overall System Design

2.1.1 Short description of the main parts of the system

- Main Parts
  - Database View Interface
    - The main window for accessing the entire system. It is designed so that all the dialog boxes and other windows can be opened from wihin it and gives a main, tabbed layout for the tables.
    - Presented as a database window with 6 push buttons that open dialog boxes for different functions
    - the 6 push buttons are labelled:
      - Open
      - Add
      - Edit
      - delete
      - Search
      - Print
    - Each table in the database is displayed in a different tab
    - Main window displays database in a box above function buttons

* Has a menu bar at the top that provides shortcuts to the functions and an about section with a help section that contains a user manual
* Add Interface
  - Dialog box that contains forms that allow the user to enter data specific to the table they want to add an item to
  - Drop down menu to select table
  - Fields with editable text boxes/line edits to enter attributes
  - confirm button
* Edit Interface
  - Dialog box that contains forms that allow the user to edit data specific to the table they want to add an item
  - Drop down menu to select table
  - Fields with editable text boxes/line edits to enter attributes
  - confirm button
* Delete Interface
  - Dialog box allowing the user to delete a specific item from any of the tables
  - Select table drop down menu
  - Select item drop down menu
  - Delete button
  - Delete all itmes button
  - require a password to b entered first in a dialog box
* Search Interface
  - Dialog box allowing the user to search through any or all of the tables to find a specific search term/query
  - Select table drop down menu
  - Search term QLineEdit
  - Search push button
* Print Interface
  - Dialog box allowing the user to select a type of form to print out based on selected information like a list of members, an invoice, or a regime.

- 3 Combo boxes allowing the user to select the type of form, the table they want the information from, and the information.
- A push button allowing the user to confirm they want to print the selected details.

* Password Interface
  - Dialog box that presents the user an outlet for entering the password for the system
  - Enter password lineEdit allowing the user to enter their password
  - Enter password push Button allowing the user to confirm the password they've entered

2.1.2 System flowcharts showing an overview of the complete system

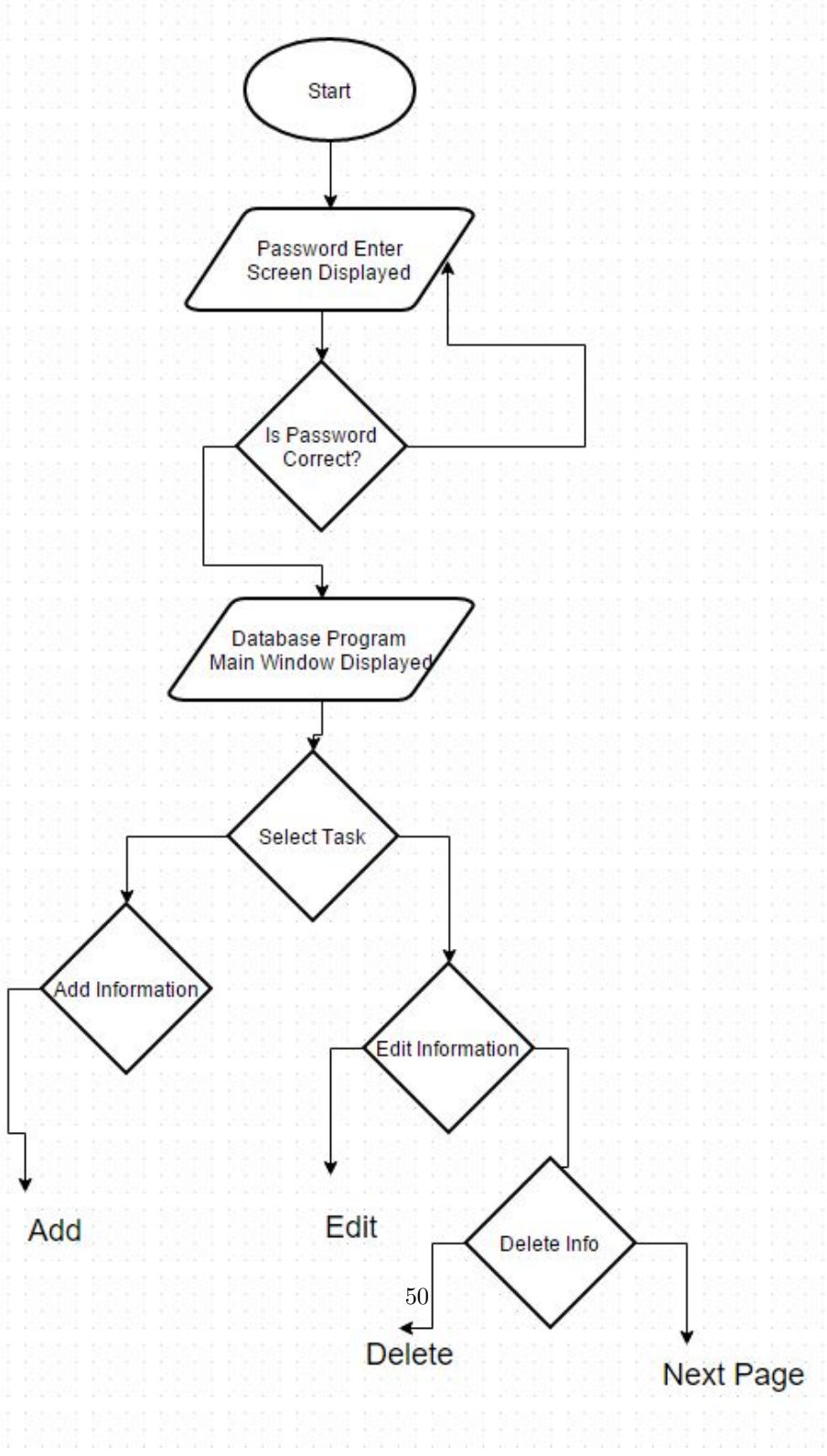


Figure 2.1: Flowchart

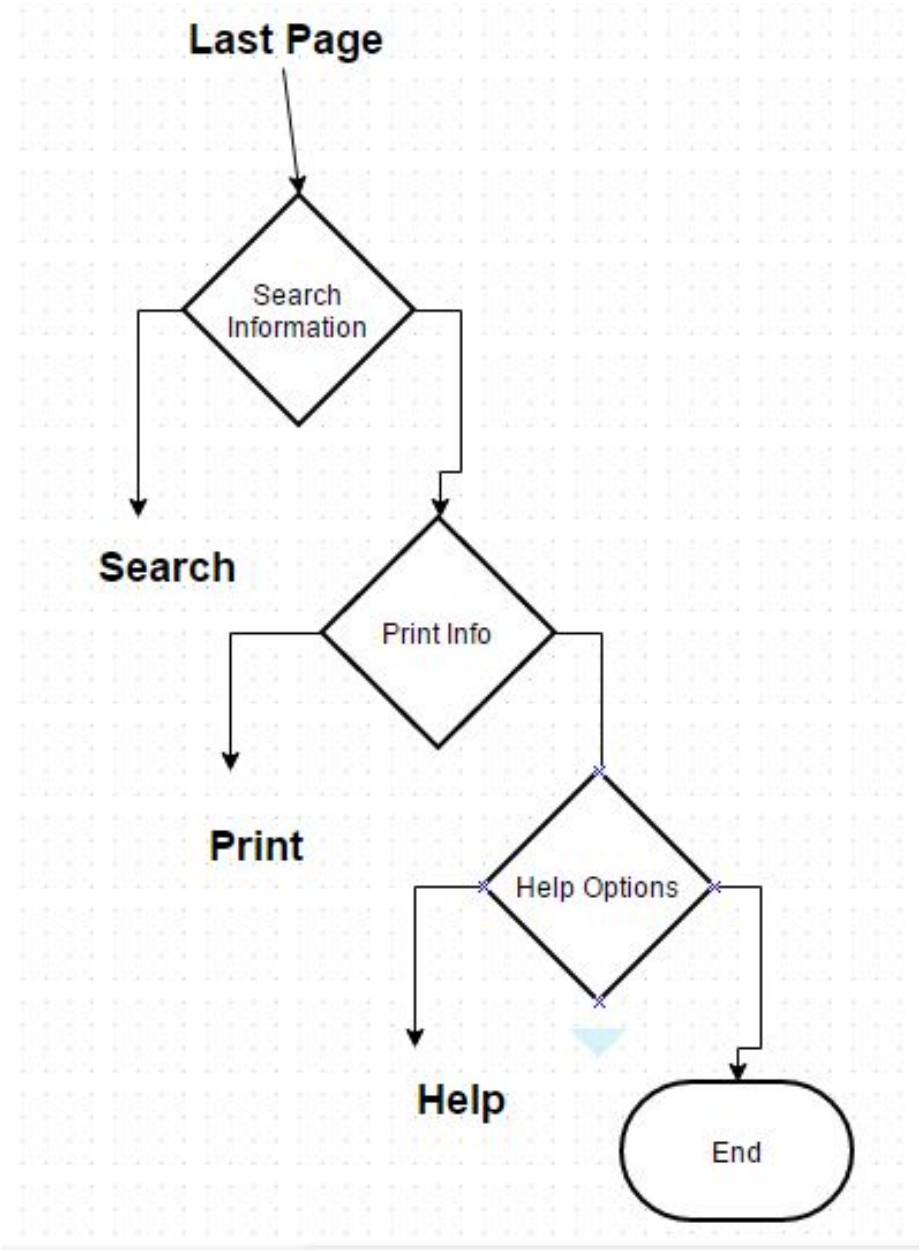


Figure 2.2: Flowchart

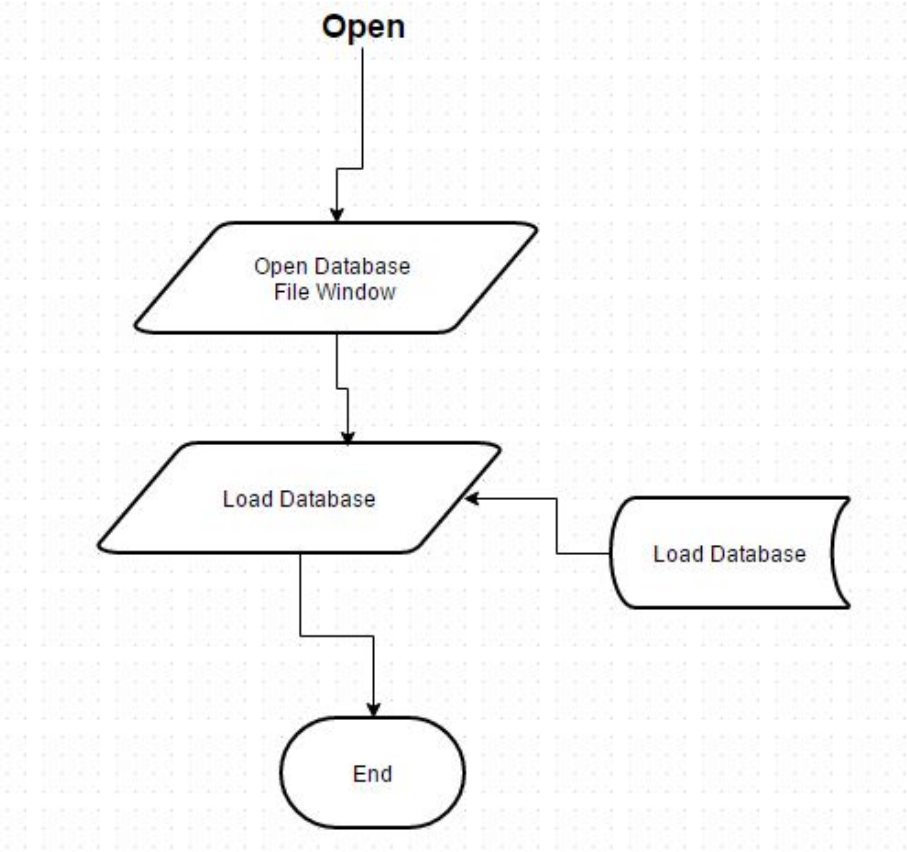


Figure 2.3: Flowchart

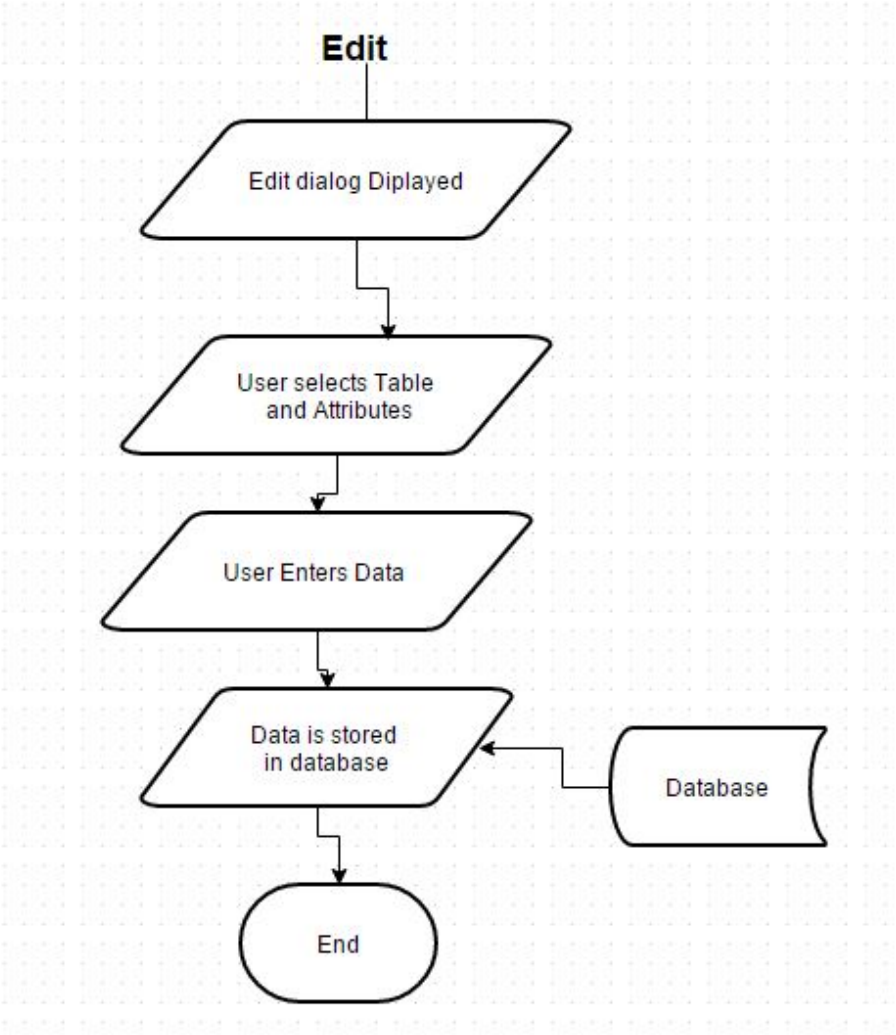


Figure 2.4: Flowchart

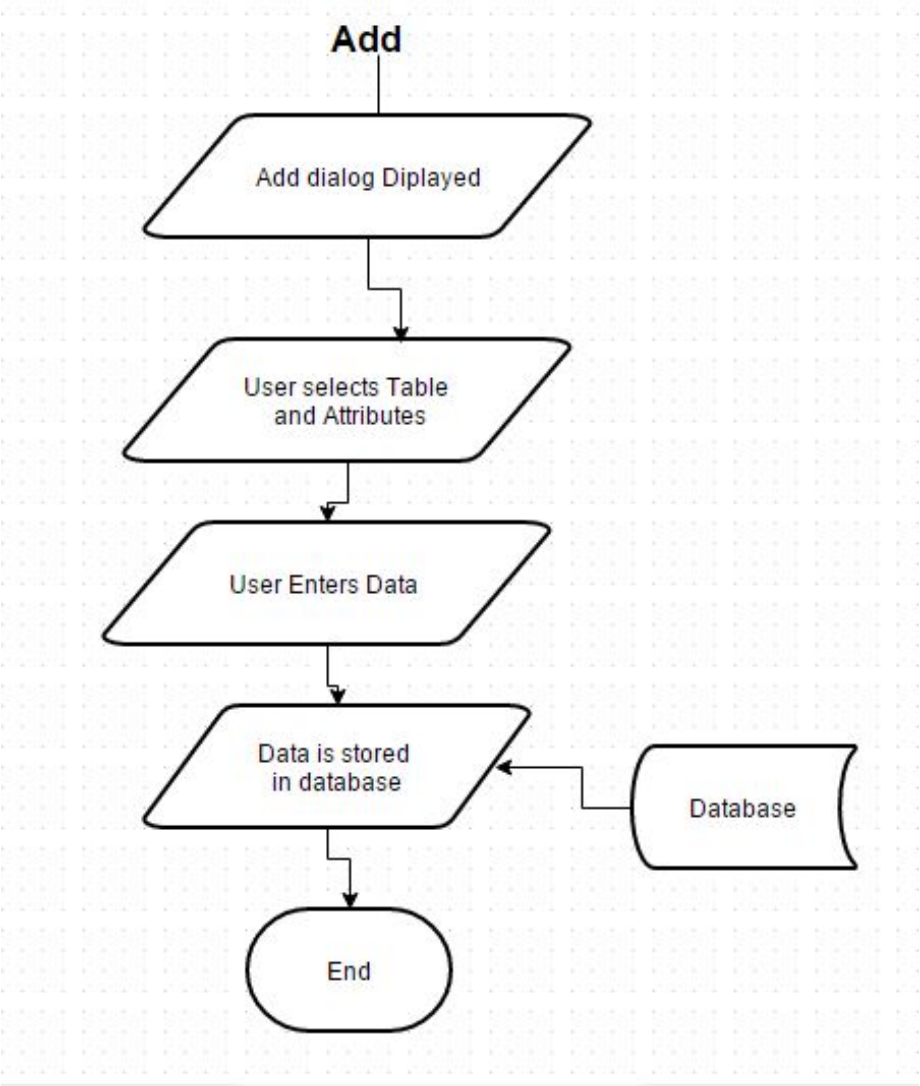


Figure 2.5: Flowchart

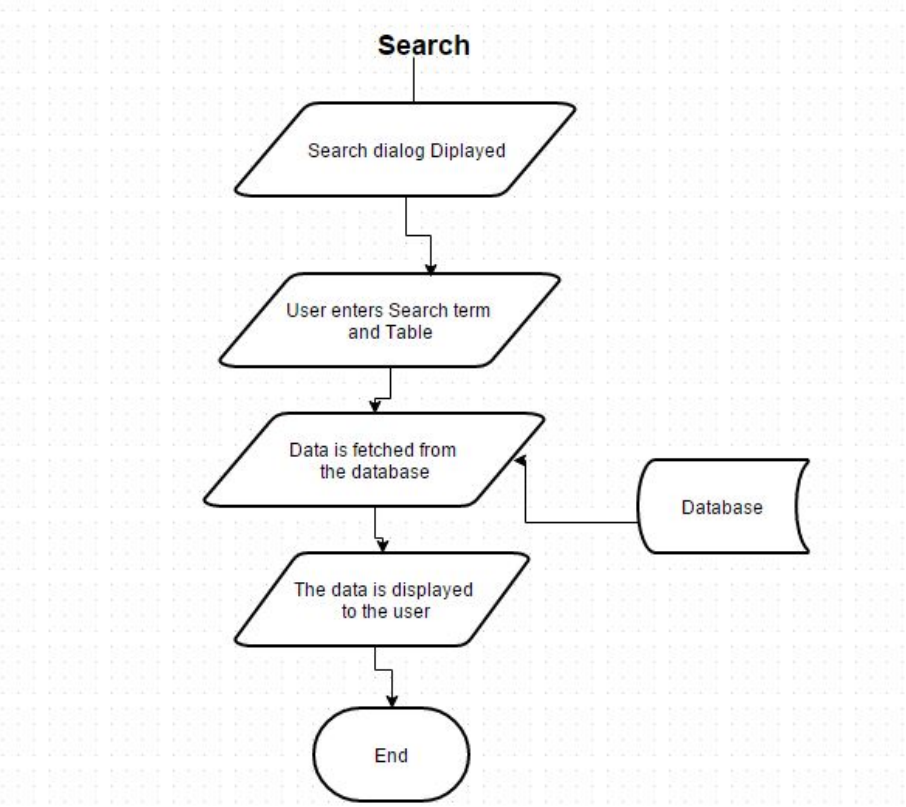


Figure 2.6: Flowchart

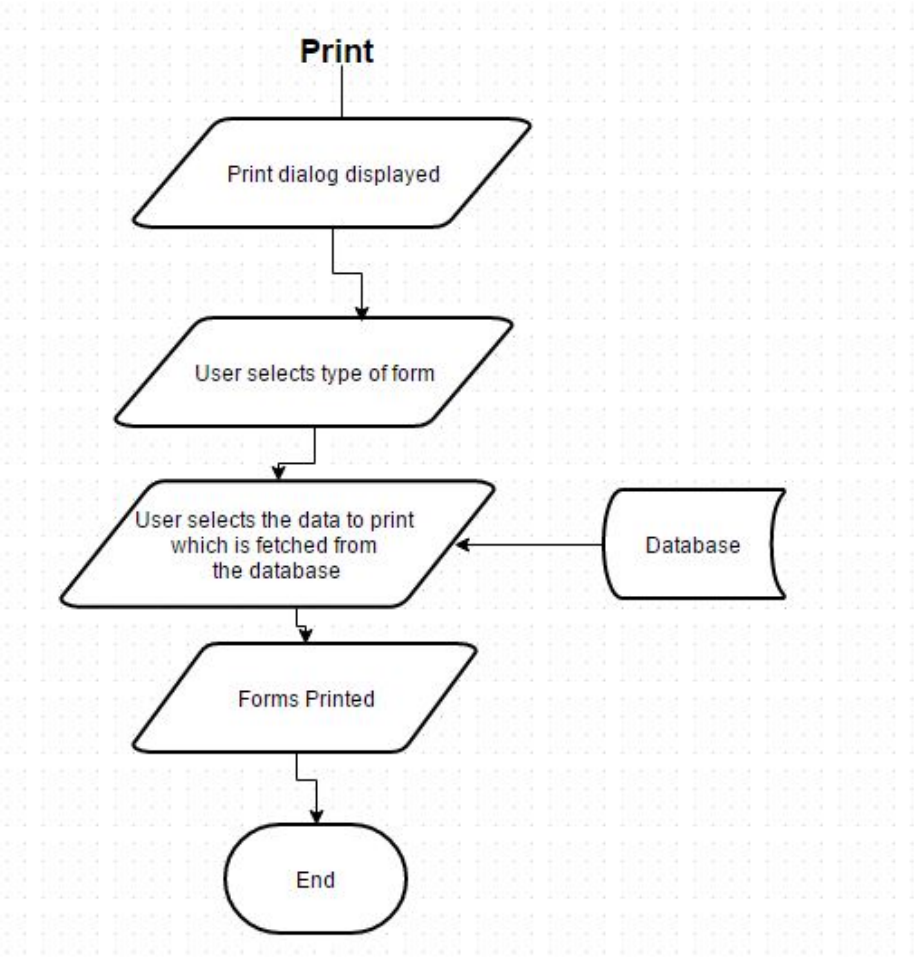


Figure 2.7: Flowchart

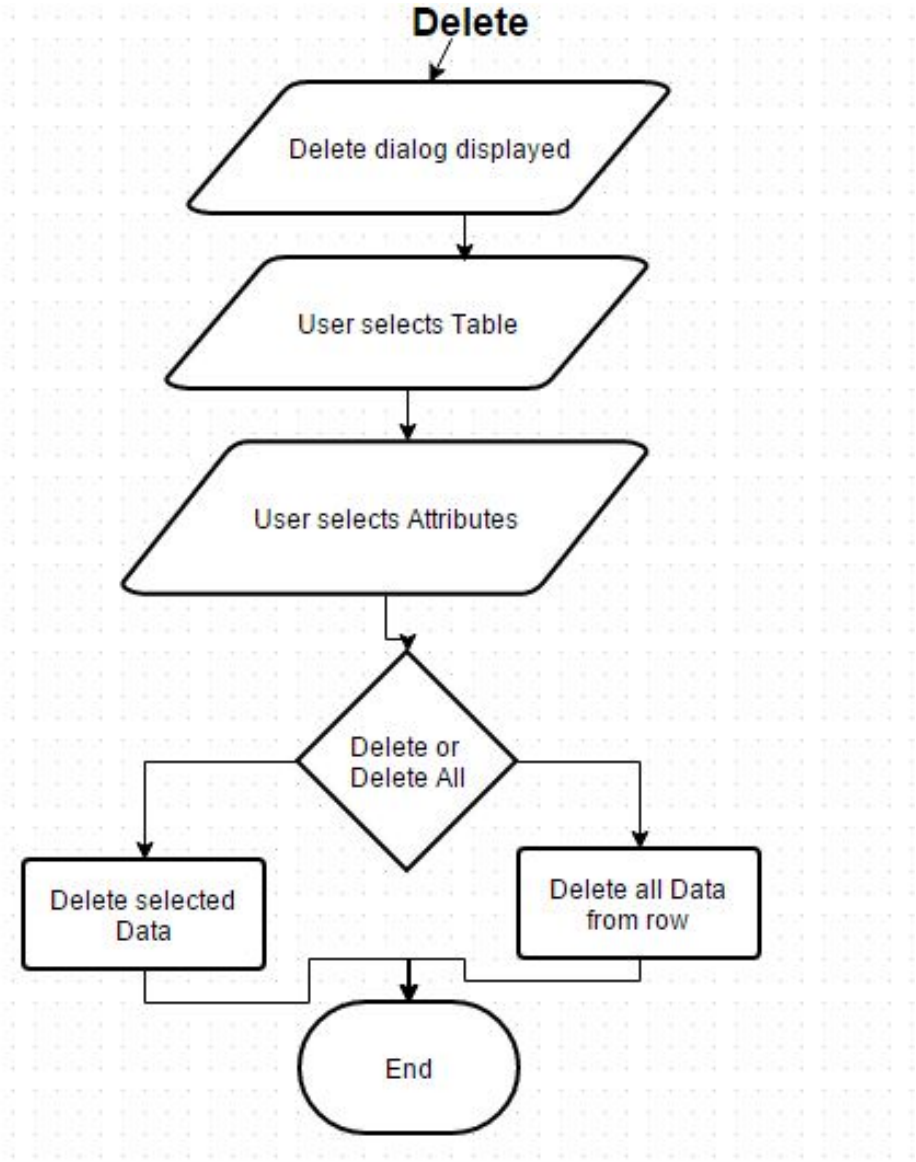


Figure 2.8: Flowchart

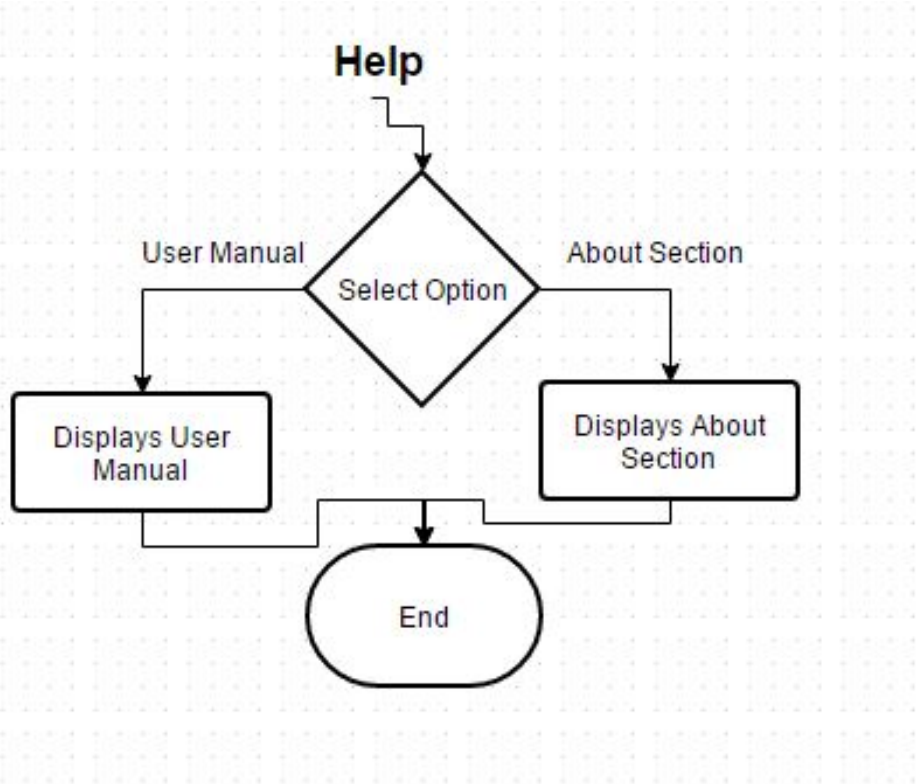
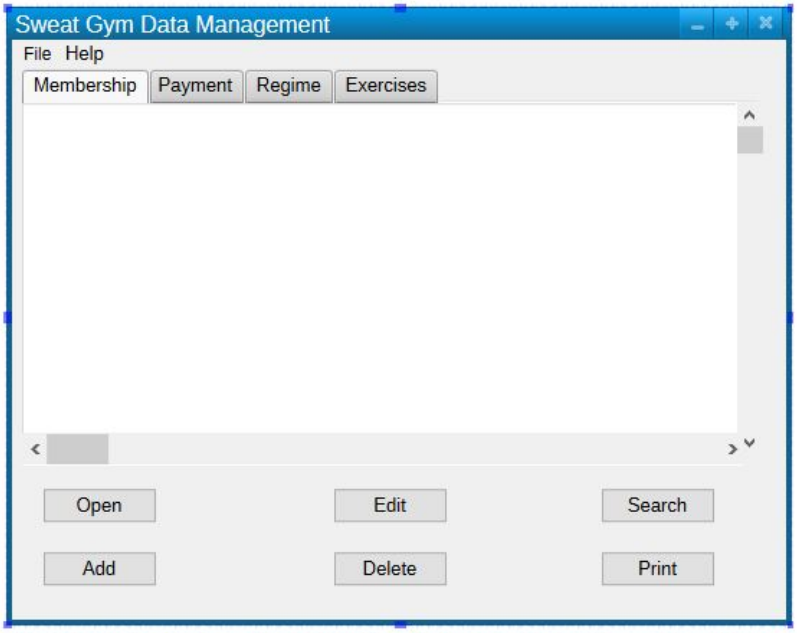


Figure 2.9: Flowchart

2.2 User Interface Designs



This is the design for the main window. The window would allow the user to get to any of the other dialogs. This includes all the programs main functions including opening the database file, adding or editing items in the database, deleting items from the database, searching through the database and printing different documents based on information from the database. It also has the main view for the tables displayed in a tabbed layout for each table. It also has shortcuts to all the functions in the file dropdown menu and links to a digital user manual and an about section in the help drop down menu.

Figure 2.10: GUI Design

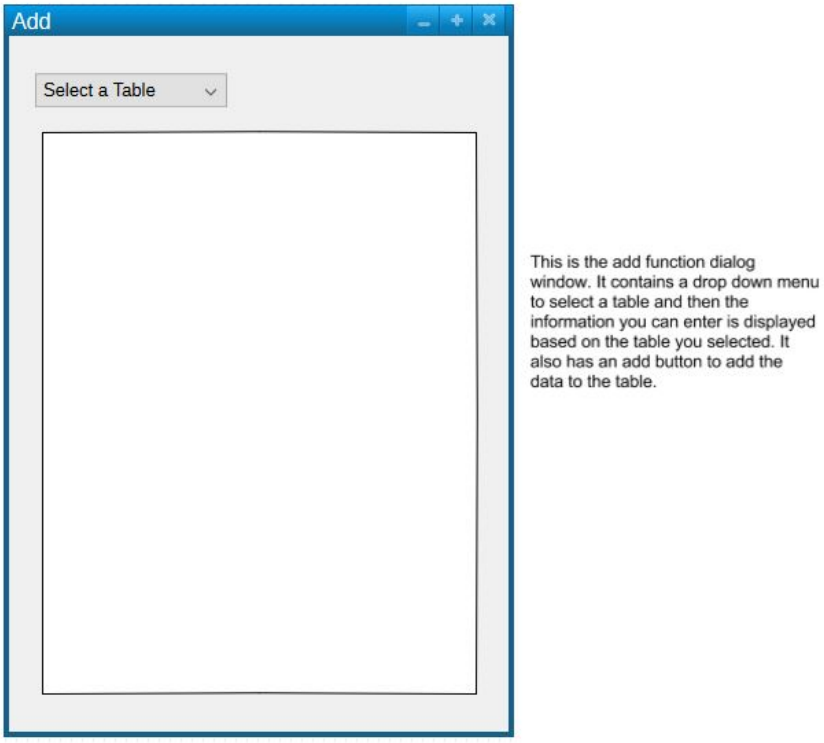


Figure 2.11: GUI Design

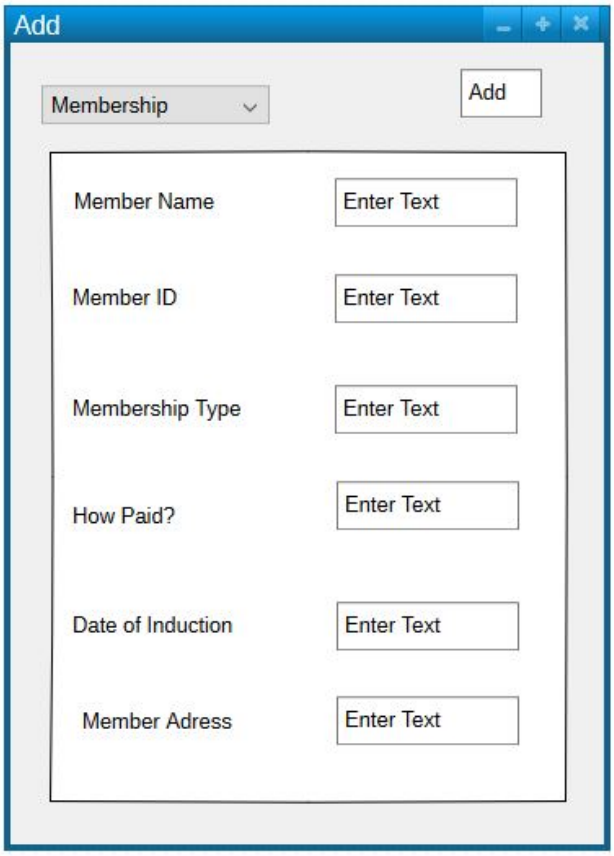


Figure 2.12: GUI Design

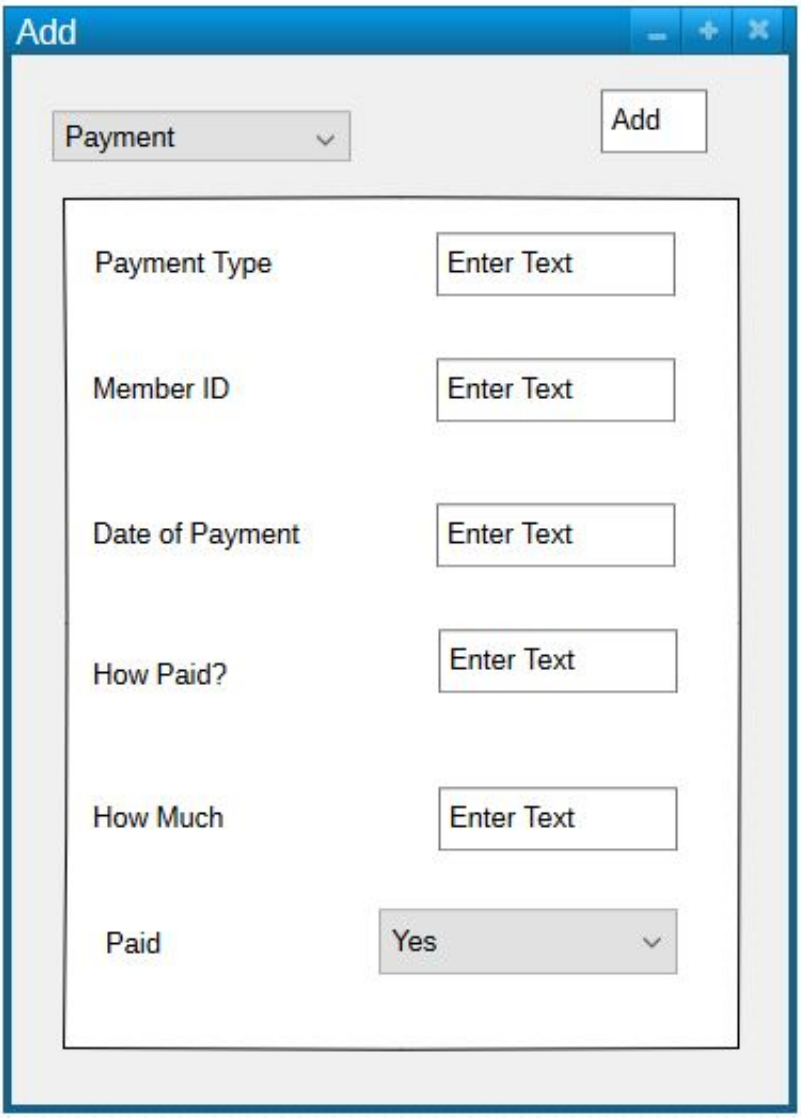


Figure 2.13: GUI Design

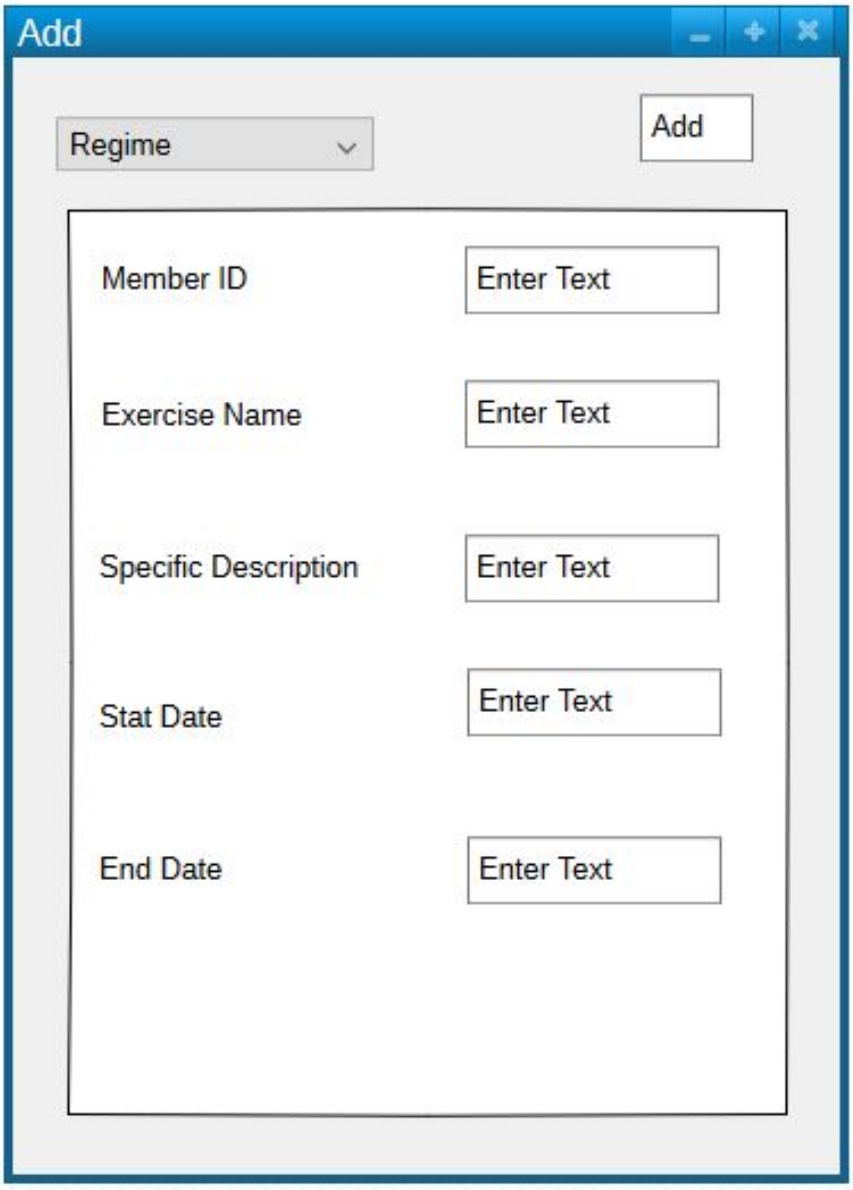


Figure 2.14: GUI Design

Figure 2.15: GUI Design

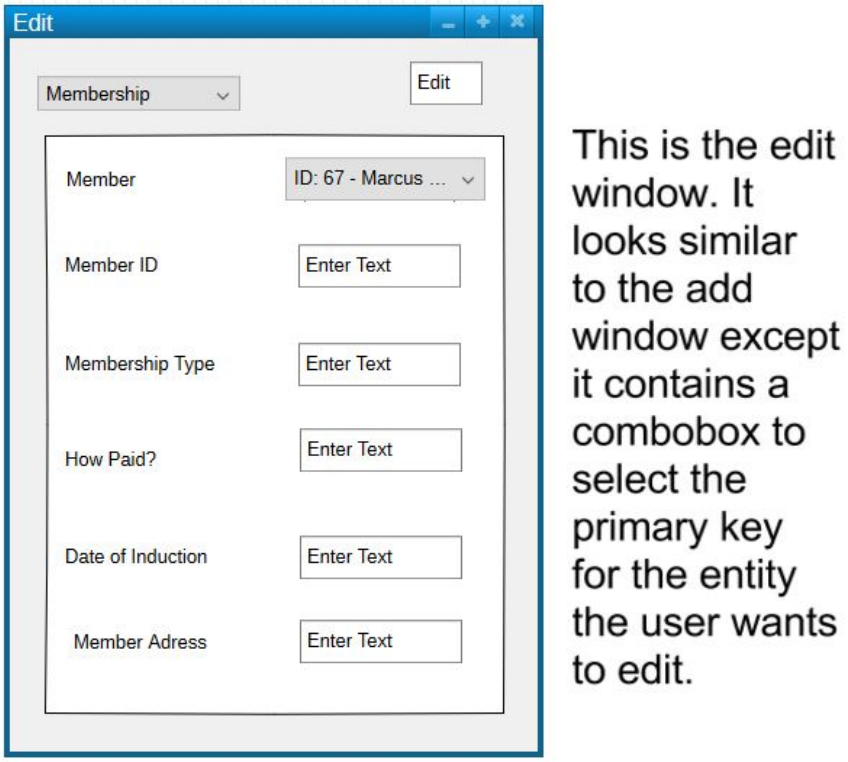
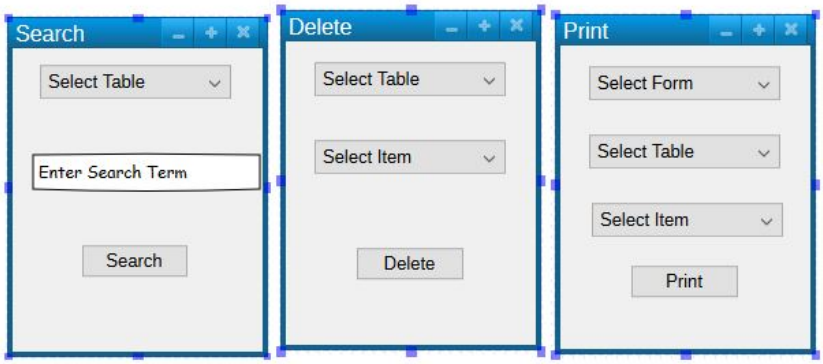


Figure 2.16: GUI Design



These are the designs for the Search, Delete and Print dialog boxes. The search Dialog box contains a combobox to select the table to search through, although it does contain an option to search through all of the tables. It contains a line edit for the search term and a push button to execute the search. The delete dialog box contains 2 comboboxes to select the table and select the item. The is an option for all items in the combobox. There's a delete push button to execute the function but it opens up a new dialog for a password to be entered so an inexperienced user won't accidentally delete any data. The print dialog contains 3 comboboxes to select the form you want to print and the specific data you want in that type of form. It can be executed by the print push button.

Figure 2.17: GUI Design

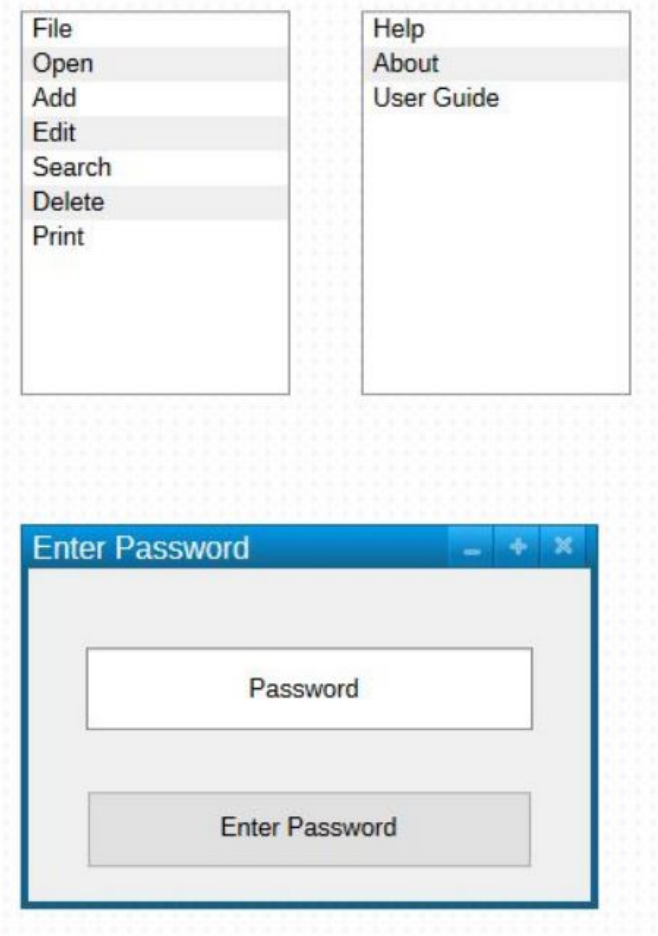


Figure 2.18: GUI Design

2.3 Hardware Specification

The owner currently uses a Notebook laptop powered by an intel 1st generation i5 processor and 4GB of RAM. He already uses this laptop to run his current system which uses more resources and power than this system will hopefully require, but if not he still has more than enough horse power to run the new

system. His use of a laptop is convienient as it means his system is portable so he doesn't need to do any data entry (if required) confined to his office and away from his client if he doesn't want too and it also means he has a battery so in the case of a power outtage he wont lose any data. Though it is worth noting that he may soon be upgrading to a more powerful desktop soon so while this lowers portability, he has more power for the proposed system and he can easily have a battery backup in the form of a UPS(Uninterptable Power Supply). All members will be running this program off the same workstation so the databases will be stored locally. This wont be a problem as the laptop has a 500 gb hard drive that wont be filled up too quickly and has lots of room for databases and the program. The program will be ouput through a display with a resolution of at least 600 * 800 (like the one used in the laptop) to accomodate the amount of screen real estate the program will require and the program will be operated and have data inputed using mouse/trackpad and keyboard (also contained within my clients laptop). The system will not use any additional peripherals. The system will also need to run on a windows os (windows 7, 8, 8.1, or 10) as thats what its been developed on and intended to run on, though it may also be compatable with MacOS and many Linux distributions as python is fairly portable and compatable with all of those languages. Though its worth noting that the program will only be compiled to work on a windows pc as thats the operating system used by my client.

2.4 Program Structure

2.4.1 Top-down design structure charts

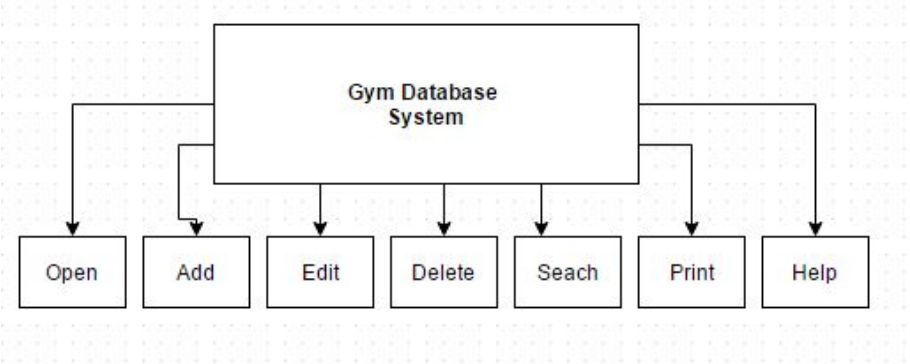


Figure 2.19: Structure Chart

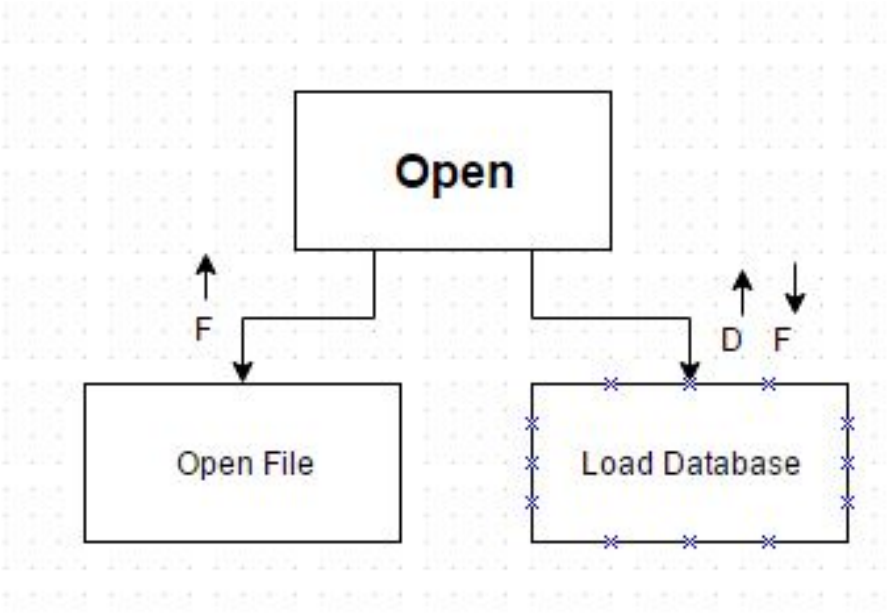


Figure 2.20: Structure Chart

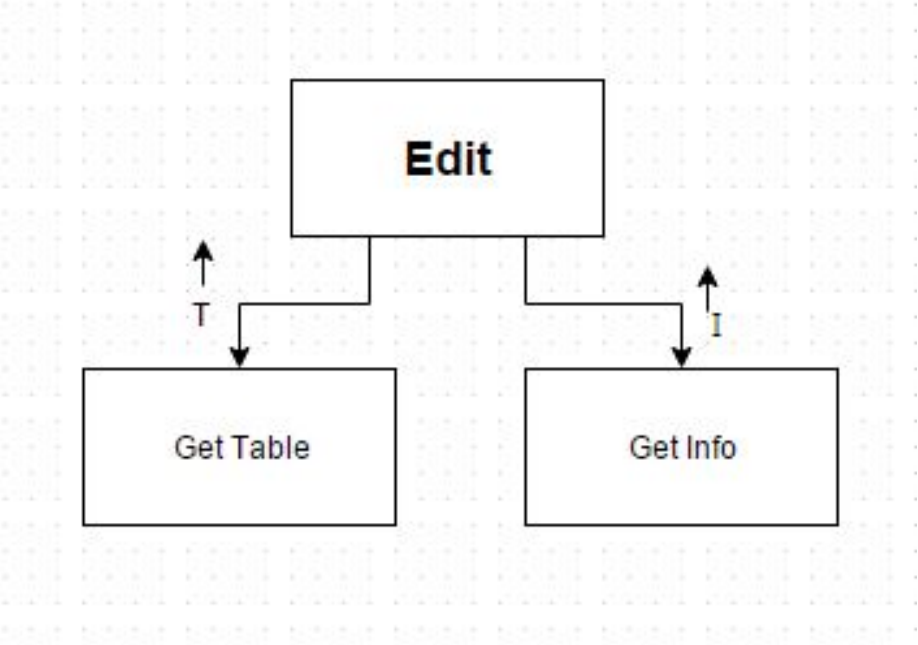


Figure 2.21: Structure Chart

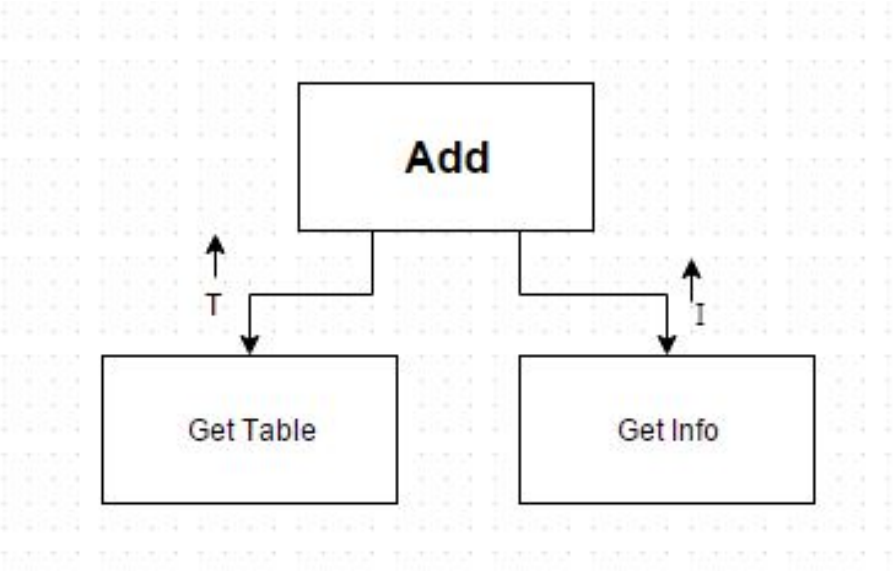


Figure 2.22: Structure Chart

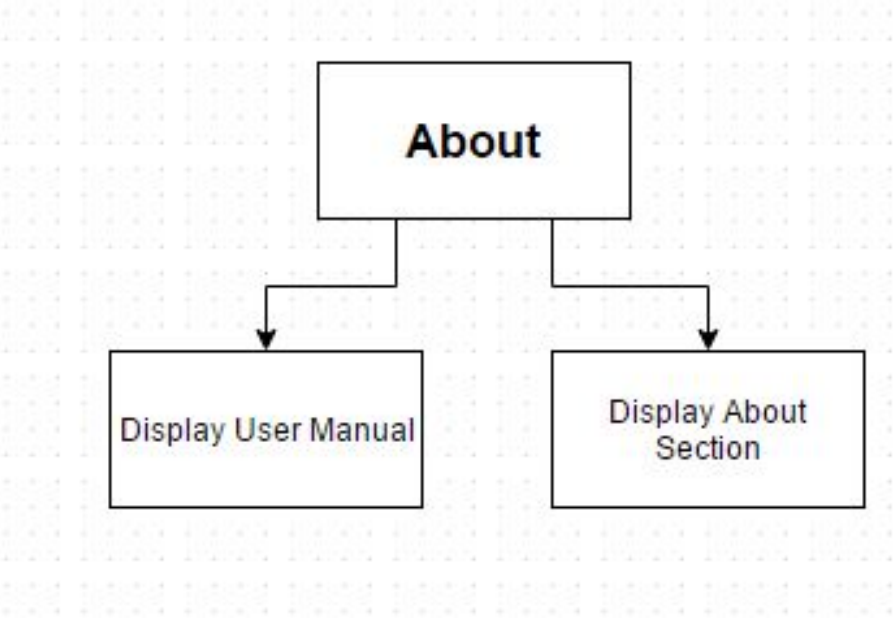


Figure 2.23: Structure Chart

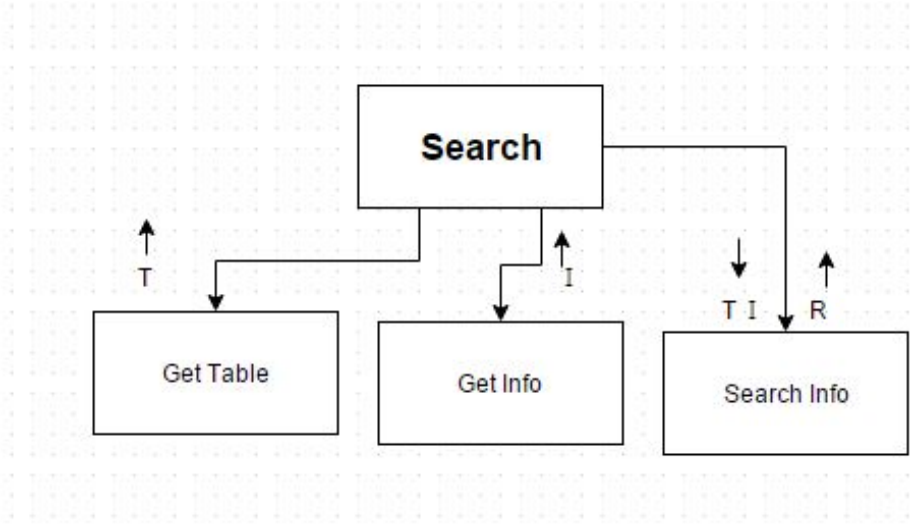


Figure 2.24: Structure Chart

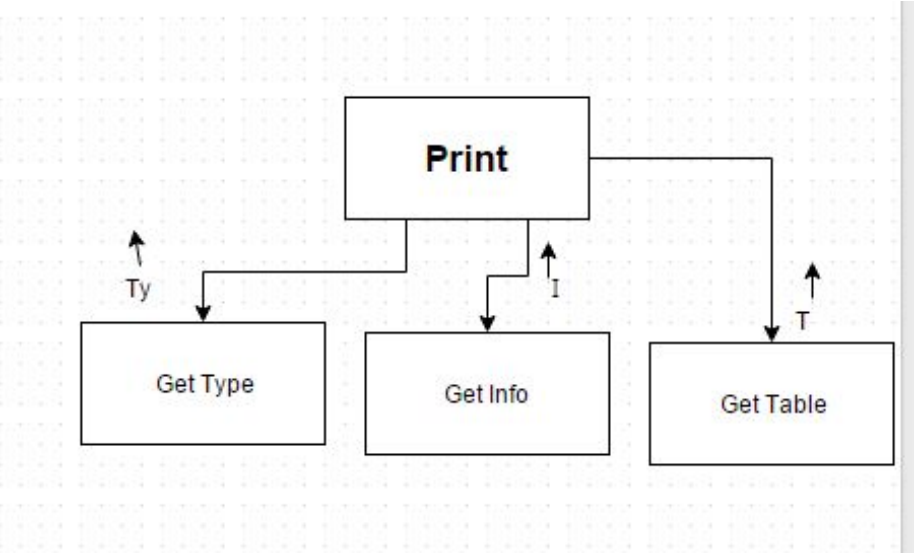


Figure 2.25: Structure Chart

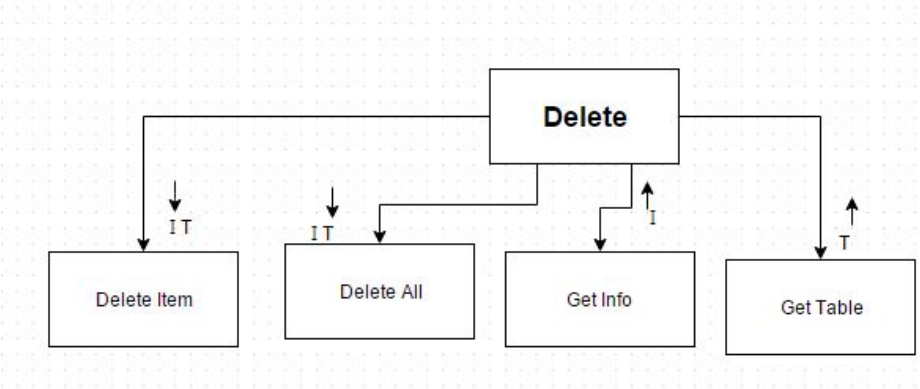


Figure 2.26: Structure Chart

2.4.2 Algorithms in pseudo-code for each data transformation process

```
1
2  START
3
4  Function FetchTotalForInvoice (database, person)
5      sql connect database as db
6      db.cursor <--- """Select How Much Where MemberID
           = "105" Payments"""
7      data <--- cursor.execute
8      total <-- 0
9      for amount in data:
10         total <--- total + data
11     return total
12 END
13
14 START
15
16 Function GenerateMemberID(database)
17     sql connect database as db
18     db.cursor <--- """Select Count (Distinct
           MemberID) From Members"""
19     HighestPlace <--- cursor.execute
20     MemberID <---- HighestPlace +1
21     return MemberID
22
23 END
24
```

```
START

Function GenerateExerciseID(database)
    sql connect database as db
    db.cursor <--- """Select Count (Distinct
        ExerciseID) From Exercises"""
    HighestPlace <--- cursor.execute
    ExerciseID <---- HighestPlace +1
    return ExerciseID

END
```

All the other instances of data transformation in my program are accurately represented in the SQL Queries section of this coursework as my other data transformation processes are all done primarily in SQL and don't require any pseudocode.

2.4.3 Object Diagrams

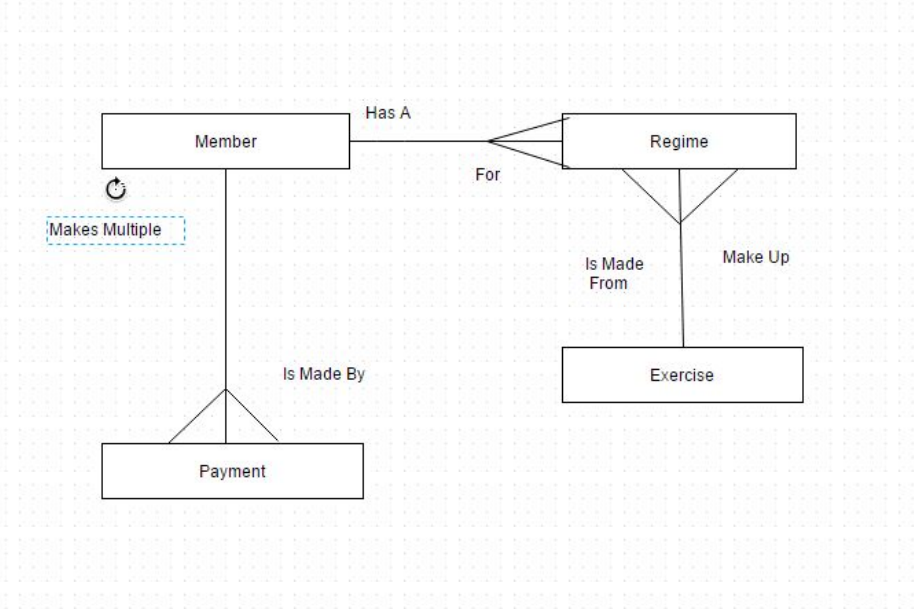


Figure 2.27: Relationship Diagram

2.4.4 Class Definitions

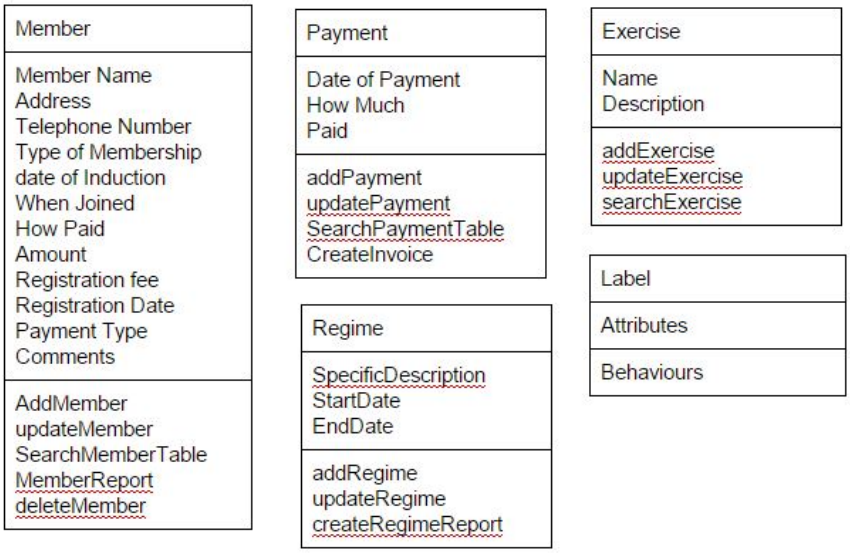


Figure 2.28: Class Definitions

2.5 Prototyping

For the prototype I created various functional GUI's to test how my program will look, feel and operate for my client and serveral commandline python functions to test some of the functionality. This allowed me to make sure the program was coming along to my clients expectations.

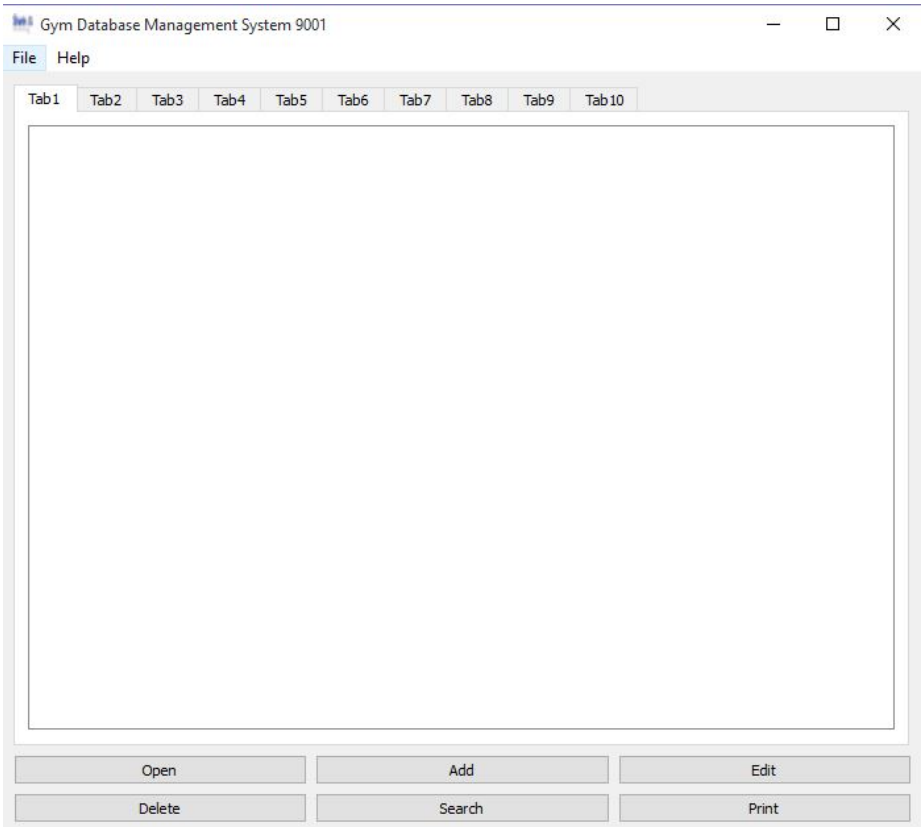


Figure 2.29: Prototype Main Gui Window

This shows the window for my main Gui. The blank white space is where the database will go once the program is complete, with each tab representing a different table. My client used this and seemed pleased with the potential of the programm but felt like he needed some functionality to get an idea of how the program will work.

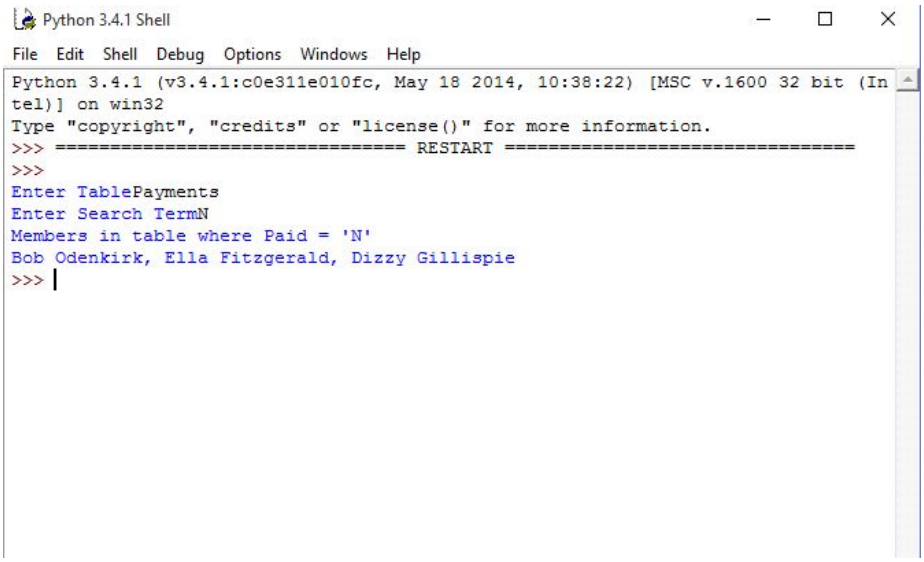


Figure 2.30: Prototype Search Commandline Window

This shows a commandline interface for a search function I made to show the potential of the functionality of the program for my client. He seemed impressed by the potential problems this search function could solve for when he needs to find any specific information quickly like how in this example he could find out everyone who had outstanding payments for membership.

2.6 Definition of Data Requirements

2.6.1 Identification of all data input items

- Member Name
- Member Address
- Member Telephone Number
- Type of Membership
- Join Date
- Date of Induction
- How Paid
- Amount Paid

- Registration Fee
- Registration Date
- Payment Type
- Date of Payment
- How Much
- Paid?
- Exercise Name
- Exercise Description
- Specific Description
- Start Date
- End Date

2.6.2 Identification of all data output items

- Member Details Printout
- Invoice Printout
- Memberlist Printout
- Member Regime

Output to Database

- Member Name
- Member Address
- Member Telephone Number
- Type of Membership
- Join Date
- Date of Induction
- How Paid
- Amount Paid
- Registration Fee
- Registration Date
- Payment Type
- Date of Payment

- How Much
- Paid?
- Exercise Name
- Exercise Description
- Specific Description
- Start Date
- End Date

2.6.3 Explanation of how data output items are generated

Output	How the Output is Generated
Member Name	Input by the user
Member Address	Input by the user
Member Telephone Number	Input by the user
Type of Membership	Input by the user
Join Date	Input by the user
Date of Induction	Input by the user
How Paid	Input by the user
Amount Paid	Input by the user
Registration Fee	Input by the user
Registration Date	Input by the user

Figure 2.31: Output Explanations

Payment Type	Input by the user
Date of Payment	Input by the user
How Much	Input by the user
Paid?	Input by the user
Exercise Name	Input by the user
Exercise Description	Input by the user
Specific Description	Input by the user
Start Date	Input by the user
End Date	Input by the user
Member Details Printout	Fetched from the Member and Payment Details Database
Invoice Printout	Fetched from the Member and Payment Details Database
Memberlist Printout	Fetched from the Member Details Database
Member Regime	Fetched from the Regime Database

Figure 2.32: Output Explanations

2.6.4 Data Dictionary

Entity	Attribute	Data Type	Length	Constrain	Description
Member	Member No	Int	(10)	Primary Key	The unique membership ID
Member	Name	String	(30)	Not Null	The name of the member
Member	Address	String	(30)	Not Null	The members address
Member	Telephone Number	String	(10)	Not Null	The members preferred contact number
Member	Type of Membership	String	(10)	Not Null	The type of membership
Member	Join Date	Datetime	(8)	Not Null	The date the member joined
Member	Date Of Induction	DateTime	(8)	Not Null	The date of the members induction
Member	How Paid	String	(10)	Not Null	How the member payed for their membership
Member	Amount	int	(2)	Not Null	How much the member payed for

Figure 2.33: Data Dictionary

					their membership
Member	Registration Fee	int	(2)	Not Null	How much does the member have to pay for registration
Member	Registration Date	Date/Time	(10)	Not Null	The date of retgistration
Payment Details	Member No	Int	(10)	Primary Key	The unique membership ID
Payment Details	Payment Type	String	(22)	Not Null	How the client pays
Payment Details	Date Of Payment	Datetime	(8)	Not Null	The Date the payment is due
Payment Details	How Much	Int	(2)	Not Null	How much the client needs to pay
Payment Details	Paid	Boolean	(1)	Not Null	Whether the client has payed
Regime	Member No	Int	(10)	Primary Key	The unique membership ID
Regime	ExerciseID	int	(10)	Not Null	The unique exercise identifier
Regime	Specific Description	String	(50)	Not Null	More specific description of the exercise

Figure 2.34: Data Dictionary 2

					tailored for the member
Regime	Start Date	Datetime	(8)	Not Null	The date the member started the regime
Regime	End Date	Datetime	(8)	Not Null	The date the member finished doing the regime
Exercise	ExerciseID	int	(10)	Primary Key	The unique exercise identifier
Exercise	Name	String	(20)	Not Null	The name of the exercise
Exercise	Description	String	(50)	Not Null	Description of the exercise

Figure 2.35: Data Dictionary 3

2.6.5 Identification of appropriate storage media

The system only needs to be accessed by the one workstation in my clients gym meaning that the database files can be stored locally as he won't need to access the files on any other system. Although in saying that the files will be backed up in the gym owners dropbox account so that if the system the databases are stored on breaks then the most recent version of the files will be retrievable and then the files can be accessed on a different system in case of an emergency, adding a certain level of security.

2.7 Database Design

2.7.1 Normalisation

ER Diagrams

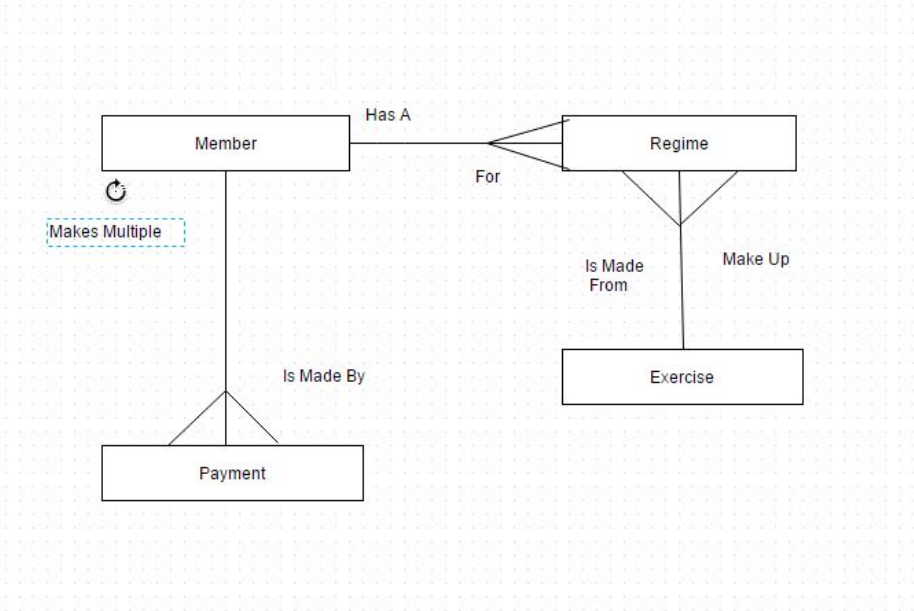


Figure 2.36: ER Diagram

Entity Descriptions

Membership Details(**Membership No.**, Last Name, First Name, Address, Telephone No., Type Of Membership, Date of Induction, When Joined, How Paid,amount, registration fee, Registration Date, Payment Type, Comments)

Payment Details(**Membership No.**,**Date of Payment**, How Much, Paid)

Regime(**Membership No.**,**ExerciseID**,Specific Description, **Start Date**, End Date)

Exercise(**ExerciseID**, Name, Description)

1NF to 3NF

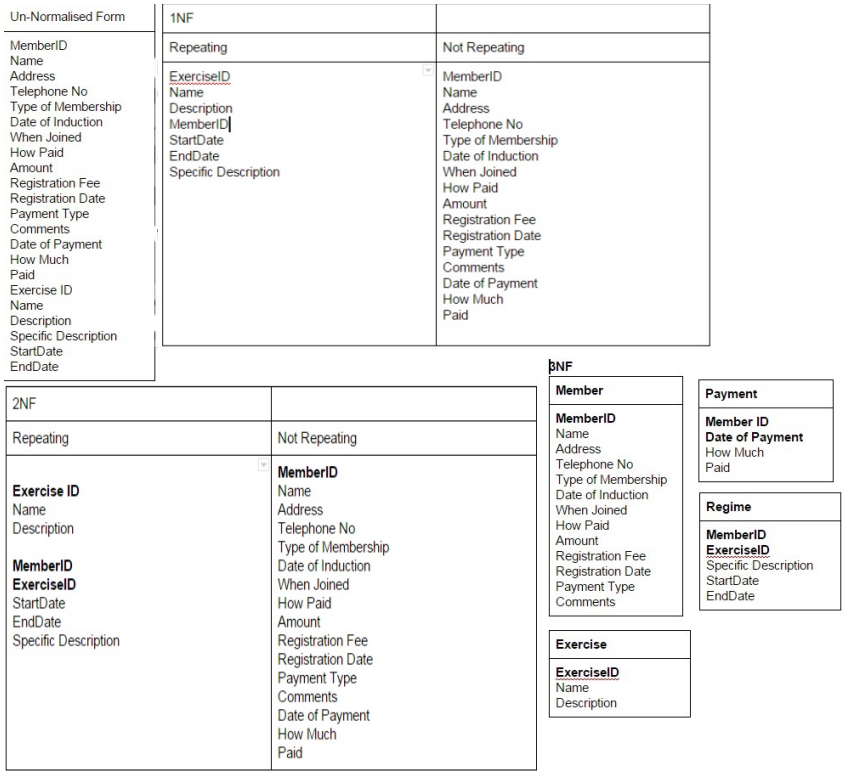


Figure 2.37: Normalisation

2.7.2 SQL Queries

SQL QUERY	DESCRIPTION
"""insert into Members(MemberID,MemberName,Address Telephone No,Type of Membership,Date of Induction,When Joined,How Paid,Amount,Registration Fee,Registration Date,Payment Type,Comments) values ('{0}','{1}','{2},'{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}','{11}'.'{12}') """.format(MemberID,MemberName,Address Telephone No,Type of Membership,Date of Induction,When Joined,How Paid,Amount,Registration Fee,Registration Date,Payment Type,Comments)	**Add a new member to the Members Table. The information is acquired from an input form made in the gui. This input form contains fields for the user to enter the input that is then passed into this query under the variable names** MemberName,Address Telephone No,Type of Membership,Date of Induction,When Joined,How Paid,Amount,Registration Fee,Registration Date,Payment Type,Comments **and one variable named** MemberID **that is autogenerated by the program.**
"""insert into Payments(MemberID, Date of Payment, How Much, Paid) values ('{0}','{1}','{2}','{3}') """.format(MemberID, Date of Payment, How Much, Paid)	**Add a new payment to the Payments Table. The information is acquired from an input form made in the gui. This input form contains fields for the user to enter the input that is then passed into this query under the variable names** MemberID, Date of Payment, How Much, Paid.
"""insert into Exercises(ExerciseID, Name, Description) values ('{0}','{1}','{2}') """.format(ExerciseID, Name, Description)	**Add a new exercise to the Exercise Table. The information is acquired from an input form made in the gui. This input form contains fields for the user to enter the input that is then passed into this query under the variable names** Name, Description **and one variable named** ExerciseID **that is autogenerated by the program.**
"""insert into Regimes(MemberID,ExerciseID,Specific Description,StartDate,EndDate) values ('{0}','{1}','{2}','{3}','{4}') """.format(MemberID,ExerciseID,Specific Description,StartDate,EndDate)	**Add a new regime to the Regime Table. The information is acquired from an input form made in the gui. This input form contains fields for the user to enter the input that is then passed into this query under the variable names** MemberID,ExerciseID,Specific Description,StartDate,EndDate.
"""updateMembers set Name = '{0}'	**Edit the details of a member in the Members Table. The information is**

Figure 2.38: SQL Queries

Address =' {1}' Telephone No =' {2}' Type of Membership = '{3}' Date of Induction = '{4}' When Joined = '{5}' How Paid = '{6}' Amount = '{7}' Registration Fee = '{8}' Registration Date = '{9}' Payment Type = '{10}' Comments = '{11}' WHERE MemberID = '{12}' """.format((MemberName,Address Telephone No,Type of Membership,Date of Induction,When Joined,How Paid,Amount,Registration Fee,Registration Date,Payment Type,Comments,MemberID)	**acquired from an input form made in the gui. This input form contains fields for the user to enter the input that is then passed into this query under the variable names** MemberName,Address Telephone No,Type of Membership,Date of Induction,When Joined,How Paid,Amount,Registration Fee,Registration Date,Payment Type,Comments. **The forms also contains a combo box which displays all the members names, accompanied by their** memberID's **to allow the user to select which member they want to edit the information of.**
"""update Payments set Paid = '{0}' How Much =' {1}' WHERE MemberID = '{2}' and Date of Payment = '{3}' """.format((MemberID, Date of Payment, How Much, Paid)	**Edit the details of a payment in the Payments Table. The information is acquired from an input form made in the gui. This input form contains fields for the user to enter the input that is then passed into this query under the variable names** How Much and Paid. **The forms also contains 2 combo boxs which displays all the members names, accompanied by their** memberID's **and their payments organised by** Date of Payment **to allow the user to select which member and month they want to edit the information of.**
Name = '{0}' Description =' {1}' WHERE ExerciseID = '{2}' """.format(Name,Description,ExerciseID)	**Edit the details of an exercise in the Exercise Table. The information is acquired from an input form made in the gui. This input form contains fields for the user to enter the input that is then passed into this query under the variable names** Name and Description. **The forms also contains a combo box which displays all the Exercises names, accompanied by their** exerciseID's **to allow the user to select which exercise they want to edit the information of.**

Figure 2.39: SQL Queries

"""update Regime set Specific Description = '{0}' Start Date = '{1}' End Date = '{2}' WHERE MemberID = '{3}' and ExerciseID = '{3}' """.format((Specific Description, Start Date, End Date, MemberID, ExerciseID)	**Edit the details of a regime in the Regime Table. The information is acquired from an input form made in the gui. This input form contains fields for the user to enter the input that is then passed into this query under the variable names** Specific Description, Start Date and End Date. **The forms also contains 2 combo boxs which displays all the members names, accompanied by their** memberID's **and their exercises organised by their** ExerciseID **to allow the user to select which member and regime they want to edit the information of.**
"""delete from '{0}' where '{1}' = '{2}' """.format(Members,ColumnMemberID,MemberID)	**Delete an item in a table. The information is acquired from an input form made in the gui. The input form contains 3 comboboxes allowing the user to select the table, the column, and the row.**
"""select from Members where MemberID = '{0}' """.format(MemberID)	**Search for a specific item in a table. The information is acquired from an input form made in the gui. The input form contains a combobox allowing the user to select the table and a lineEdit allowing the user to enter the search term. This data is then passed into this query.**
"""select Name from Members"""	**Fetches the information for a form, in this case a list of all the members in the Member Table. The kind of form and data passed depends on the information input by the user using the print interface, which has several comboboxes allowing the user to select the kind of form and the data used.**

Figure 2.40: SQL Queries

2.8 Security and Integrity of the System and Data

2.8.1 Security and Integrity of Data

The system will have data entered through combo boxes/drop down menus where possible to avoid the user entering bad data but for a large amount of the fields (like a members name) has to be a raw input typed by the user with a keyboard. All the data input will be checked for errors. e.g making sure the data type is correct - strings can't be entered where an integer is required.

2.8.2 System Security

The system will be protected behind a password that is only known by certain users/members of staff at the gym and a secondary password will be required to use certain functions like the function to delete any of the data or edit anything important.

2.9 Validation

Item	Example	Validation/Verification	Comments
Password	password2	Lookup Check	Makes sure that they password entered is the correct one that's been preset by the user
Name	Cab Calloway	Presence Check	Makes sure somethings entered
Adress	2, Fake Street	Presence Check	Makes sure somethings entered
Telephone No	01353 669878	Presence Check	Makes sure somethings entered
Date of Induction	11/05/2015	Presence Check Type Check	Makes sure somethings entered and it's in a date time format
When Joined	14/05/2015	Presence Check Type Check	Makes sure somethings entered and it's in a date time format
Amount	50	Presence Check Type Check	Makes sure somethings entered and it's an integer

Figure 2.41: Validation

Registration Fee	50	Presence Check Type Check	Makes sure somethings entered and it's an integer
Registration Date	12/05/2015	Presence Check Type Check	Makes sure somethings entered and it's in a date time format
Comments	Has Asthma	Presence Check	Makes sure somethings entered
Date of Payment	12/05/2015	Presence Check Type Check	Makes sure somethings entered and it's in a date time

Figure 2.42: Validation

			format
How Much	50	Presence Check Type Check	Makes sure somethings entered and it's an integer
Specific Description	10 Push Ups	Presence Check	Makes sure somethings entered
StartDate	12/06/2015	Presence Check Type Check	Makes sure somethings entered and it's in a date time format
EndDate	12/09/2015	Presence Check Type Check	Makes sure somethings entered and it's in a date time format
Name	Push Ups	Presence Check	Makes sure somethings entered
Description	Vertical Lifts .etc....	Presence Check	Makes sure somethings entered

Figure 2.43: Validation

2.10 Testing

Outline Plan

Test Series	Purpose of Test Series	Testing Strategy	Strategy Rationale
1	Test the flow and navigation between the different menus and interfaces included in the GUI.	Top-Down Testing	
2	Test the validation methods for the data input by the user	Bottom-Up Testing	Each component will be tested after development.
3	Test that all the input data is stored in the correct place	Black Box Testing	
4	Test algorithms to make sure the output is correct	White Box Testing	
5	Test that the system meets the specification.	Acceptance Testing	

Figure 2.44: Outline Plan

Detailed Plan

Test Series and Number	Purpose	Description	Test Data	Test Data Type	Expected Result	Actual Result	Evidence in Appendix
1.1	Test that the password enter dialog responds correctly to the wrong password	The password dialog should reopen prompting the user to try again	" " - Leave the password lineEdit blank and then click the enter button	Erroneous	The password dialog will reopen		
1.2	Test that the password responds correctly to the correct password	The password dialog should close opening the main interface	"password" - Enter the correct password and then click the enter button	Normal	The main interface will open after the password dialog closes		

Figure 2.45: Detailed Plan

1.3	Test that the open button works correctly	The windows file explorer should open to allow the user to select the right database file	Click the open button	Normal	The windows file explorer should open while the main interface remains open		
1.4	Test that the open shortcut in the toolbar	The windows file explorer should	Click the open button	Normal	The windows file explorer should		

Figure 2.46: Detailed Plan

	works correctly	open to allow the user to select the right database file			open while the main interface remains open		
1.5	Test that the table view in the main interface changes to accommodate the opened database	The table view should change to include data from the database selected with each table appearing in a different table	Open a database	Normal	The table view will now contain information stored in the database file		

Figure 2.47: Detailed Plan

1.6	Test that the Add button works correctly	The Add dialog box should open allowing the user to select and input the data they want to add to the database	Click the add button	Normal	The Add dialog box should open while the main interface remains open		
1.7	Test that the Add shortcut in the toolbar works correctly	The Add dialog box should open allowing the user to select	Click the add button	Normal	The Add dialog box should open while the main interface		

Figure 2.48: Detailed Plan

		and input the data they want to add to the database			remains open		
1.8	Test that the select table combo box works correctly in the Add dialog.	The section in the layout should fill with a form with input options based on the table selected	Change the select table combo box option to all avaibale tables	Normal	The input form should change to contain informati on and input options based on the selected table		
1.9	Test that the Add dialog responds correctly after the Add button has been pushed	The dialog box should close returning the users focus to the main interface	Enter all appropria te data and hit the Add button	Normal	The dialog box should close with the main interface still open		

Figure 2.49: Detailed Plan

1.10	Test that the Edit button works correctly	The Edit dialog box should open allowing the user to select and input the data they want to edit in the database	Click the edit button	Normal	The Edit dialog box should open while the main interface remains open		

Figure 2.50: Detailed Plan

1.11	Test that the Edit shortcut in the toolbar works correctly	The Edit dialog box should open allowing the user to select and input the data they want to edit in the database	Click the edit button	Normal	The Edit dialog box should open while the main interface remains open		
1.12	Test that the select table combo box works correctly in the Edit dialog.	The section in the layout should fill with a form with input options based on the table selected	Change the select table combo box option to all avaibale tables	Normal	The input form should change to contain informati on and input options based on the selected table		

Figure 2.51: Detailed Plan

1.13	Test that the Edit dialog responds correctly after the Edit button has been pushed	The dialog box should close returning the users focus to the main interface	Enter all appropria te data and hit the Edit button	Normal	The dialog box should close with the main interface still open		
1.14	Test that the Delete button works correctly	The Delete dialog box should open allowing	Click the Delete button	Normal	The Delete dialog box should open while the		

Figure 2.52: Detailed Plan

		the user to select the informati on they want to delete while the main interface remains open			main interface is still open		
1.15	Test that the delete shortcut in the toolbar works correctly	The Delete dialog box should open allowing the user to select the informati on they want to delete while the main interface remains open	Click the Delete shortcut	Normal	The Delete dialog box should open while the main interface is still open		

Figure 2.53: Detailed Plan

1.16	Test that the Select Table combobox in the delete dialog works correctly	The Select Item combobox should then include different options for the user to select	Select all possible options from the Select Table combo box	Normal	The Select Item combo box should fill with data based off of the selected table		
1.17	Test that	The	Click the	Normal	The		

Figure 2.54: Detailed Plan

	the Delete dialog box responds correctly to the Delete or Delete All button being clicked	dialog box should close bringing the Main Interface into the users focus	Delete and Delete All Buttons		Delete Dialog should close leaving just the Main Interface Open		
1.18	Test that the Search Button works correctly	The search Dialog should open allowing the user to enter the information they wish to search for	Click on the Search button	Normal	The Search dialog should open while the Main Interface stays open in the background		

Figure 2.55: Detailed Plan

1.19	Test that the Search shortcut in the toolbar works correclty	The search Dialog should open allowing the user to enter the informati on they wish to search for	Click on the Search Shortcut	Normal	The Search dialog should open while the Main Interface stays open in the backgrou nd		
1.20	Test that the Search button in	This should display the	Enter all appropria te informati	Normal	A new Results Dialog should		

Figure 2.56: Detailed Plan

	the Search dialog works correctly	results of the search for the user	on and click the Search button		open closing the Search dialog but keep the Main Interface open in the backgrou nd		
1.21	Test that the Close button in the result dialog functions correctly	The Results dialog should close drawing the user's focus to the Main Interface again	Click on the Close Button	Normal	The Results dialog should close while the Main Interface window is bought back to the user's attention		

Figure 2.57: Detailed Plan

1.22	Test the Print Button works correctly	The Print Dialog should open so that the user can select the form and informati on they want to print	Click on the Print Button	Normal	The Print Dialog should open leaving the Main Interface open in the Backgrou nd		
1.23	Test that the combobo xes in the print function work	The combobo xes should change the informati	Select all the different options in all 3 combobo xes	Normal	The Combob oxes should have different options		

Figure 2.58: Detailed Plan

	correctly	on in the subseque nt combobo x when an item is selected in one of them			based on what's entered in the other combobo xes		
1.24	Test that the Print button in the Print Dialog works correctly	The Print dialog should close bringing the Main Interface back into the user's focus	Click on the Print Button	Normal	The Print Dialog should close leaving only the Main Interface Open		

Figure 2.59: Detailed Plan

1.25	Test that the User Manual Option in the toolbar works	A digital version of the User Manual should open in another window so that the user can easily identify how to perform an operation within the program	Click on the User Manual Shortcut	Normal	The User Manual should open up in a dialog box leaving the Main Interface Open		
1.26	Test that the About option in the toolbar works	The about section should open so that the	Click on the About option	Normal	A Dialog box containin g informati on about		

Figure 2.60: Detailed Plan

		user can identified the programs current version, creator and any other necessar y informati on			the program should be opened while the Main Interface remains open in the program		

Figure 2.61: Detailed Plan

2.1	Verify that the password entered is correct	The password will match the one stored inside the program	The correct password The incorrect password	Normal Erroneous	The password will be correct and the program will carry on as so The program will not continue until the correct password is entered and the user will be prompted as such		
2.2	Verify that the file opened is a database (.db) file	Opening the wrong type of file should result in an error	A Database File A Word	Normal Erroneou	The program will continue and load the database The		

Figure 2.62: Detailed Plan

			File	s	program will prompt the user with an error		
2.3	Verify that the informati on entered in the Add input form is of the correct type	If the data is not the correct type the user should receive an error message	Appropri ate Data Incorrect data type(s)	Normal Erroneou s	The program will continue with the data The user will receive an error message and be prompted to change the necessar y data		

Figure 2.63: Detailed Plan

2.4	Verify that the informati on entered in the Edit input form is of the correct type	If the data is not the correct type the user should receive an error message	Appropri ate Data Incorrect data type(s)	Normal Erroneou s	The program will continue with the data The user will receive an error message and be prompted to change the necessar y data		

Figure 2.64: Detailed Plan

2.5	Verify that the search term entered in the Search Dialog actually yields results	If the search comes up without results the user should receive a message saying so	A correct search term An incorrect search term	Normal Erroneous	The program continues on with the search results The user receives a message and is prompted to use a different search term		
3.1	Verify that all Member details that are input are added to the database	All the details should be added to the correct place in the database	Member information	Normal	The data should be added into the appropriate place in the database		

Figure 2.65: Detailed Plan

3.2	Verify that all Payment details that are input are added to the database	All the details should be added to the correct place in the database	Payment informati on	Normal	The data should be added into the appropria te place in the database		
3.3	Verify that all Exercise details that are input are added to the	All the details should be added to the correct place in the	Exercise informati on	Normal	The data should be added into the appropria te place in the database		

Figure 2.66: Detailed Plan

	database	database					
3.4	Verify that all Regime details that are input are added to the database	All the details should be added to the correct place in the database	Regime information	Normal	The data should be added into the appropriate place in the database		
3.5	Verify that all Member details that are input are updated in the database	All the correct details should be updated in the database	Member information	Normal	The data should be edited into the appropriate place in the database		
3.6	Verify that all Payment details that are input are updated in the database	All the correct details should be updated in the database	Payment information	Normal	The data should be edited into the appropriate place in the database		

Figure 2.67: Detailed Plan

3.7	Verify that all Exercise details that are input are updated in the database	All the correct details should be updated in the database	Exercise information	Normal	The data should be edited into the appropriate place in the database		
3.8	Verify that all Regime details that are	All the correct details should be updated	Regime information	Normal	The data should be edited into the appropria		

Figure 2.68: Detailed Plan

	input are updated in the database	in the database			te place in the database		
3.9	Verify that all Member details that are input are deleted in the database	All the correct details should be deleted from the database	Member informati on	Normal	The data should be deleted from the appropria te place in the database		
3.10	Verify that all Payment details that are input are deleted in the database	All the correct details should be deleted from the database	Payment informati on	Normal	The data should be deleted from the appropria te place in the database		

Figure 2.69: Detailed Plan

3.11	Verify that all Exercise details that are input are deleted in the database	All the correct details should be deleted from the database	Exercise informati on	Normal	The data should be deleted from the appropria te place in the database		
3.12	Verify that all Regime details that are input are deleted in the database	All the correct details should be deleted from the database	Regime informati on	Normal	The data should be deleted from the appropria te place in the database		
4.1	Verify that the	The search	Search some	Normal	The search		

Figure 2.70: Detailed Plan

	Search function works correctly	function should allow the user to input a search term and then query that database and return the results	informati on that is known to be in the database		function should return the correct informati on		
4.2	Test that the Invoice function correctly calculate s the total money a client needs to pay	The invoice function should add together the sums of all the unpad months of a member	A member that the user already knows the total money owed of	Normal	The function should print the correct total along with other informati on		

Figure 2.71: Detailed Plan

4.3	Test that the Print function fetches the correct data	The Print function fetches data from the database to print in the forms	Any table and form that the user knows the output of	Normal	The function should retrieve the correct data		
5	Make sure that the program fulfills the specificati on laid out in the analysis and	Run through the program making sure every function and aspect	Add some informati on to the database and run through all the different methods	Normal	Program fulfills the specificati on		

Figure 2.72: Detailed Plan

	design	meets this specificati on	of data manipulat ion and view the results				

Figure 2.73: Detailed Plan