Plastering Job Management Application

Kyle Kirkby

December 5, 2014

Contents

6 Evaluation   **95**

Chapter 1

Analysis

1.1 Introduction

1.1.1 Client Identification

My client is 30 year old plasterer Dan Austin who runs his own plastering
business known as DnA Plastering. Dan mainly uses his Toshiba laptop (Dual
Core Intel with 6 GB Ram and running Windows 8 64 bit) to do basic tasks
such as social networking and receiving/sending emails.

The current system is a paper based method where he records the prices and
measurements of the plastering/screening/rendering jobs he undertakes. Dan
works in an around the Suffolk/Essex area but occasionally takes on larger
jobs further afield in places such as London or Epping. All the recording and
calculations are done by Dan himself and does not require additional
assistance in completing these tasks but is looking for a digital solution to the
organisation problems faced with the current manual paper method.

Dan is looking to introduce a computer based system to replace the current
one in order to make keeping track of jobs and pricing up new jobs easier and
more efficient. Alongside this he would like to be able to keep information on
all of his customers so he can simply search for clients' details and contact
information all in one location. He will also be able to look up the jobs that he
has done for them to make sending invoices easier and manageable.

1.1.2 Define the current system

The current system in place is a paper/notebook based system where details of
clients are stored along with prices of jobs and cost of materials needed etc.

The details of the clients include their address, phone number, email, first name and surname. The infromation about the job usually includes the measurements of what needs to be plastered along with how long it will take to complete and if he is taking any labourers to too. Calculations are often also made to work out how much to charge depending on the price he is charging per square meter. This rate often changes depending on the current economy.

Once all the calculations are made, he works out how much the materials are going to cost and also how long it will take him to complete the job. Once all these calculations and prices have been evaluated he notifies the client of the price; when the price is confirmed the job is undertaken.

Finally, Dan writes out an invoice using a standard invoice book purchased from a stationary store to inform the client of the costs and charges of the job. The current folder containing the invoices for his clients is not organised and offers another problem whereby finding information for jobs is difficult due to the inability to search quickly for any given customer.

1.1.3   Describe the problems

One of the main problems in the current system is the ability to keep valuable client data from being lost or damaged as there is only one hard copy made in a notebook. The notebook is also partial to an occasional coffee stain and would not be the case if they data is stored digitally.

Another problem with the current system is the inability to easily search through the details of all the clients to find specific phone numbers or contact details. Dan often has to find client data to get in contact with them and without an easy way to search through it all he has to go through each contact to find details needed.

Being able to generate an invoice is currently a lot harder than it needs to be as all the client information and job information can be in different locations(i.e multiple workbooks or paper etc) and hard to locate.

With the proposed computer based system, it would allow Dan to search through his clients efficiently and allow him to make backups of the valuable client and job data which is otherwise hard to acheive, due to it all being stored manually in the notebook, in the current system.

1.1.4   Section appendix

Interview with client

1. **What system are you currently using?**

I've got a Toshiba laptop with Windows 8, 6GB RAM and a dual core processor

2. **Are there any issues with the system currently being employed?**

Just organisational issues when it comes to finding clients information etc. Would be good to have a database which I could search for info with.

3. **What data do you record at the moment?**

The clients name, address, phone number, job information like measurements of the rooms etc.

4. **How often do you record data in the current system and how much data is stored each time?**

A few times a week and not much, a few lines in the workbook I use.

5. **What happens to older client information?**

Normally gets lost as and when I replace any of my books I write the details down in.

6. **Are there any storage issues when it comes to storing data manually?**

No, nothing really as I only need room for a book or two.

7. **Are changes often made to existing records of client data or job info?**

Occasionally when a client gives me a new contact number but nothing major.

8. **What is the typical routine when it comes to gaining a new client or job?**

Normally I get a call from a colleague who gives me the details about a job which I can take on or not depending on the time I have available. Sometimes clients will find my number through my website dnaplastering.co.uk or through word of mouth and will ask me to give them a quote on a job. Then I will go to the job and work out any costs involved and the time it would take to complete. Then I can arrange a time to do the job and after the job is completed I will usually write out an invoice for the client telling them about the cost for materials,labouring and profit etc.

9. **What does the client recieve in terms of invoices?**

I write out an invoice and give it to them in person after the job is completed.

10. **What format should the outputs be in within the new system?**

    It would be good to be able to print the clients invoices out with minimal effort involved and possibly email the clients the invoice.

11. **Will you need to print reports and invoices or will it be sent entirely digitally to the client via email?**

    I definitely need to print out some of the invoices as most of my work is done face to face so it is easier to give it to them in person. It might be useful to email it to them as well.

12. **Is there a security issue in regards to the data you store in the current system?**

    Not much of an issues as it is just client information but added security would be a bonus.

13. **Are there any foreseeable constraints required in the proposed system?**

    None that I am aware of.

14. **Do you have a picture in mind of what the new system could look like?**

    Not really bothered about the look of it too much anything that works as expected would be brilliant.

Signed:

1.2   Investigation

1.2.1   The current system

Data sources and destinations

There are four main data sources within the current system - The plasterer,
the client, the builders merchant and visting the clients job. A client contacts
Dan through a phone call placed to Dan's mobile. Sometimes a client may
leave Dan a voicemail message if he is too busy to answer the call at that
given moment. If this is the case then Dan will get back to the client as soon
as possible. Most of the data in the current system will come from the client
or the clients job - this data will be the job measurements and the clients
contact information. The main data destinations are the forms given to the
client i.e the quote and the invoice document.

Source	Data	Example Data	Destination
Client	Client Contact information First-name - Lastname - phoneNumber - AddrLine1 - AddrLine2 - AddrLine3 - AddrLine4 - PostCode - Email - JobType	John - Smith - 07809726812 - 15 - Crowley Road - Haverhill - Suffolk - CB90DJ - john@gmail.com - Plastering Bedroom	Appointment and Client Book.
Plasterer	Appointment Time and Place	16:00 at 15 Crowley Road, Haverhill	Client Calendar or Diary
Plasterer	Measurements of Job site and Materials that need to be purchased	4m x 5m x 3m = $60m^2$ 10 Bags of Plaster	Work Notebook
Plasterer	Quote for the work that needs doing and agree a date it can be done.	£600, 1 Day, 15th October	Quote written out on paper or agree in person.
Plasterer	Quantity of materials needed for the job	25 bags of plaster and 12m of angle beading	Builders Merchant
Builders Merchant	A price for the materials needed	£350 for the bags of plaster and angle beading	Plasterer
Plasterer	Invoice - Total cost of the job broken down - cost of parts,labouring and vat. Date of Job	£600 - £350 materials - £50 VAT - 14/08/14	Client.

Algorithms

There are four main algorithms utilised in the current system. The first is an algorithm to agree the price of the job with the client.

Algorithm 1 Agreeing a price Algorithm

1: $agreed \leftarrow false$
2: **WHILE** agreed = False
3:     **IF** "Client does not agree with quoted price" **THEN**
4: Discuss price and change quote if new price is agreed upon.
5:     **ELSE**
6:         $agreed \leftarrow true$
7: Arrange a date for the work to be started on.
8:     **ENDIF**
9: **ENDWHILE**

The second algorithm currently being used in the system is an algorithm used to see whether the work is completed.

Algorithm 2 Checking whether work is complete or not.

1: $Complete \leftarrow False$
2: **WHILE** Complete = False
3:     **IF** "Issue/problem not fixed." **THEN**
4: Check the current problem and fix issue.
5:     **ELSE**
6:         $Complete \leftarrow True$
7:     **ENDIF**
8: **ENDWHILE**
9: Create and send invoice

The third algorithm being used in the system is an algorithm used to see whether the work has been paid for completely.

Algorithm 3 Checking whether work has been paid for yet.

1: $Paid \leftarrow False$
2: **WHILE** Paid = False
3:     **IF** "Money has not been given." **THEN**
4: Send invoice and contact client
5:     **ELSE**
6:         $Paid \leftarrow True$
7:     **ENDIF**
8: **ENDWHILE**
9: Update job to paid for in book.

The fourth algorithm being used in the system is an algorithm which helps to generate a quote for the client.

Algorithm 4 Generating a quote for the client.

1: $NumberOfWorkingDaysNeeded \leftarrow$ **USERINPUT**
2: $CostOfMaterials \leftarrow$ **USERINPUT**
3: $DailyRateOfCharge \leftarrow$ **USERINPUT**
4: $QuoteCost \leftarrow NumberOfWorkingDaysNeeded * DailyRateOfCharge$
5: $QuoteCost \leftarrow QuoteCost + CostOfMaterials$

Data flow diagrams

Key

Data Store

Interface

Process
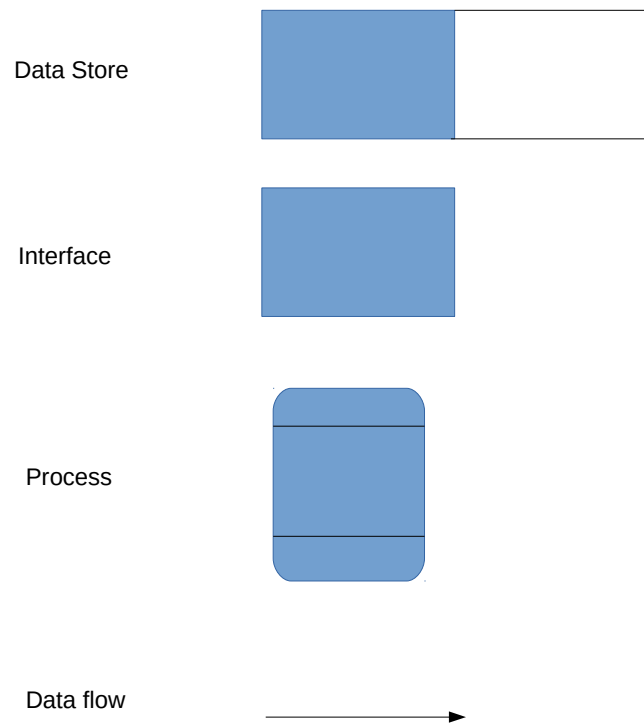
Data flow

Figure 1.1: This is the Key to be used for the following data flow diagrams.
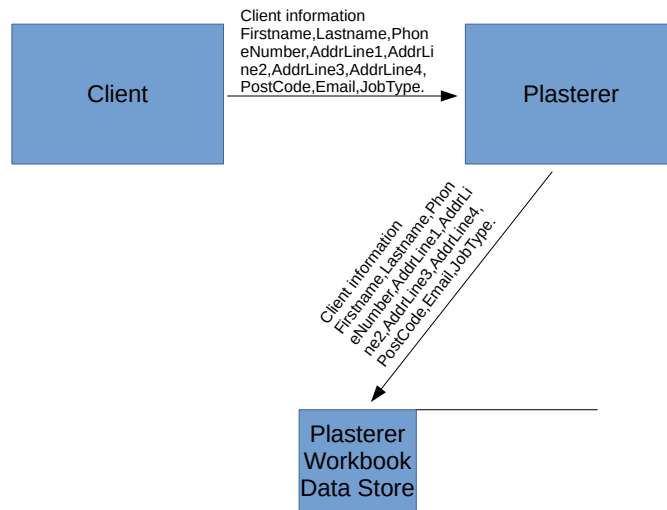
Figure 1.2: This diagram shows the flow of data when gaining a new clients information.
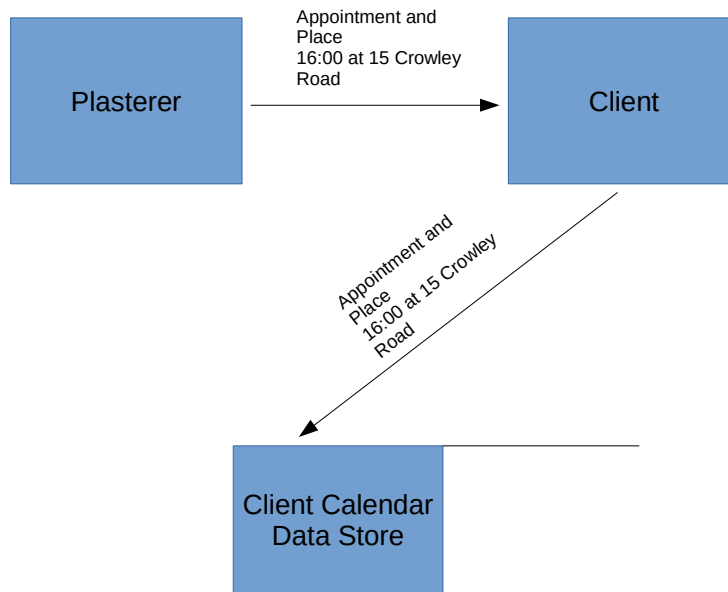
Figure 1.3: The signifies the flow of data when making an appoint for a job.
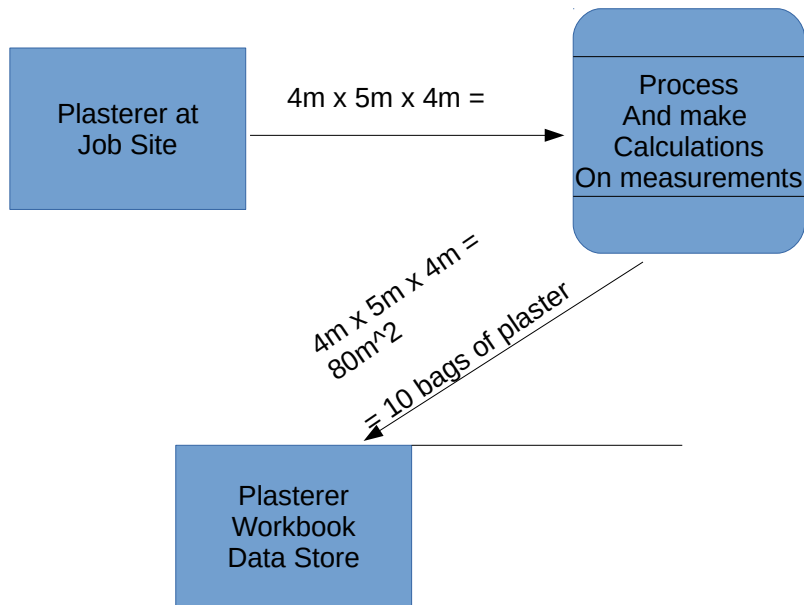
Figure 1.4: This diagram shows the flow of data when collecting the measurements for a job.
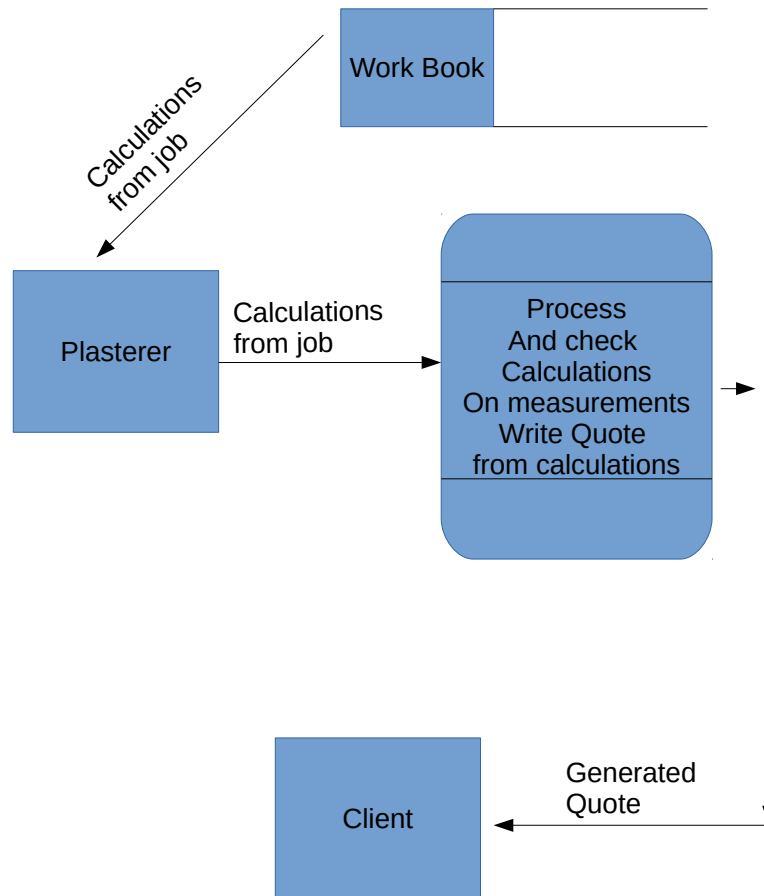
Figure 1.5: The flow of data when generating a quote for the client.
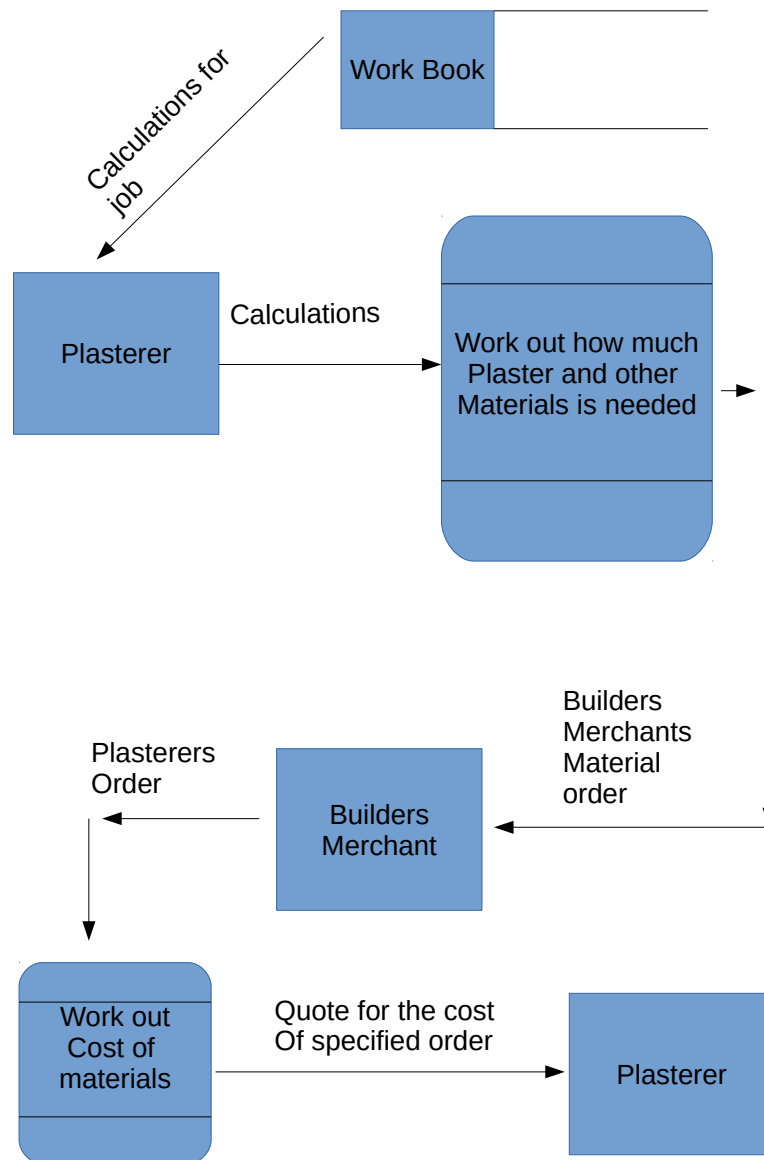
Figure 1.6: This shows the data flow when getting a quote for the materials.
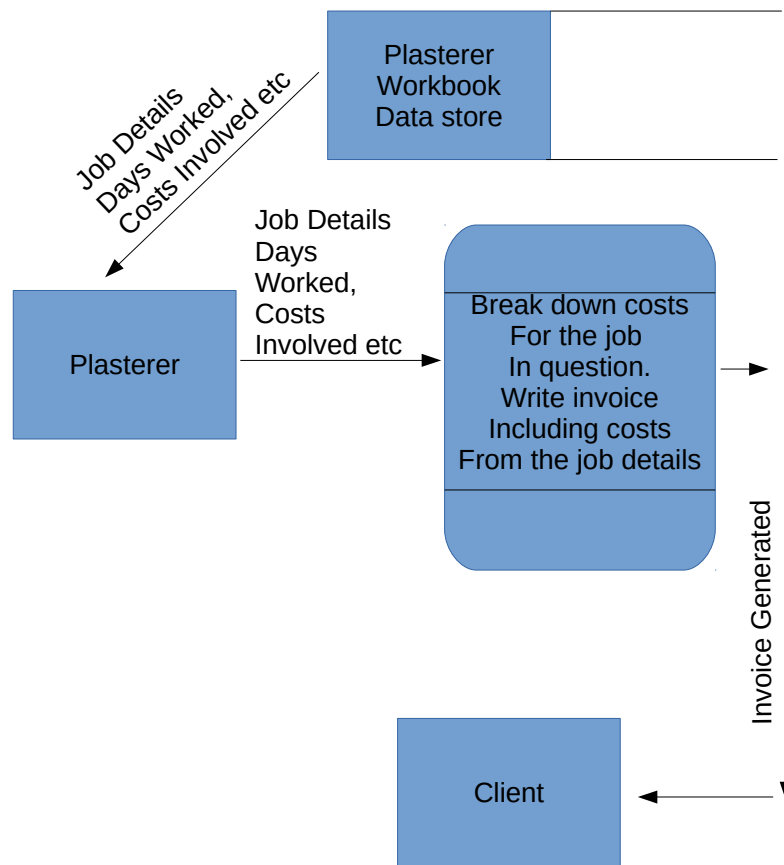
Figure 1.7: The Data flow when an invoice is given to the client.

Input Forms, Output Forms, Report Formats

In the system currently being used there are two main forms - the work book input form and the invoice output form. Below are a few examples of these forms as they are currently being used in the system:

Figure 1.8: This is an example of an input form where data is put into the system and is a page from a work book used to store details about jobs.

Figure 1.9: This is an example of an invoice output form which is given to clients.

Figure 1.10: Here is an illustration of the type of information which goes on an invoice to the client.
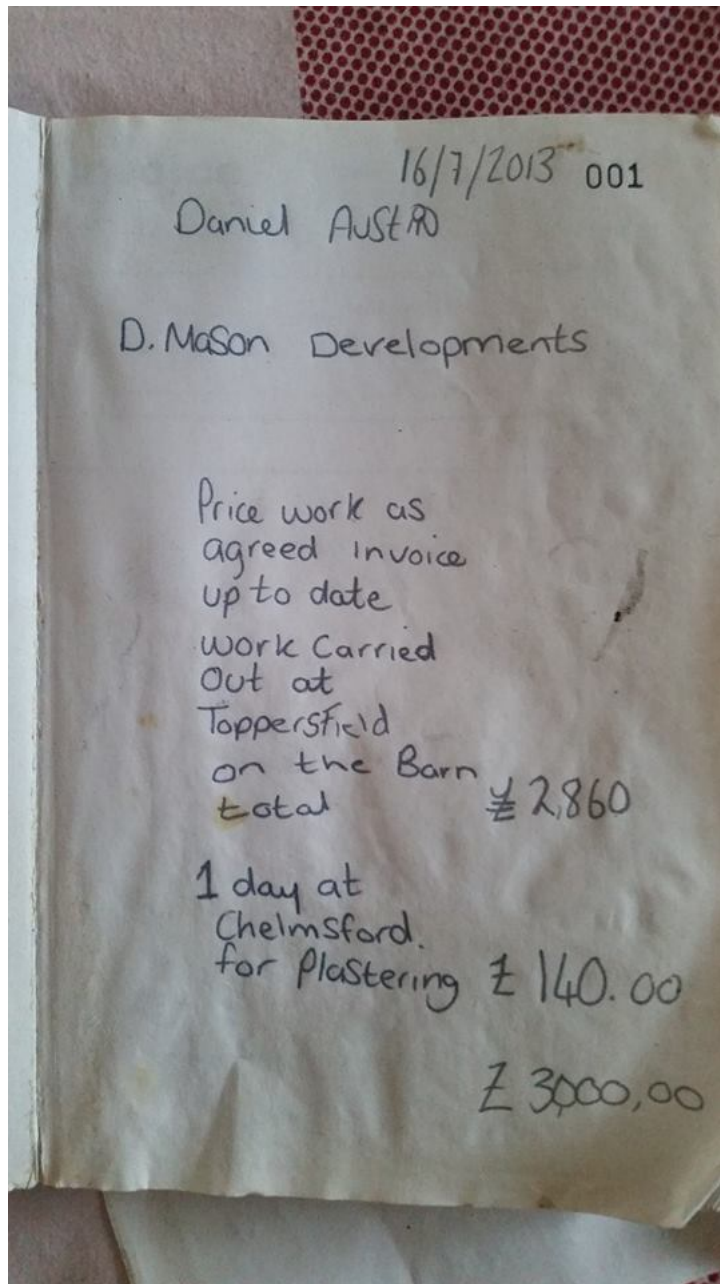
Figure 1.11: This is also another example of an invoice which is given to a client.

1.2.2   The proposed system

Data sources and destinations

Source	Data	extbfData Type	Destination
Client	Firstname	String/text	Plasterer
Client	Surname	String/text	Plasterer
Client	AddrLine1	String/text	Plasterer
Client	AddrLine2	String/text	Plasterer
Client	AddrLine3	String/text	Plasterer
Client	AddrLine4	String/text	Plasterer
Client	PostCode	String/text	Plasterer
Client	Email	String/text	Plasterer
Client	MobNumber	String/text	Plasterer
Plasterer	ClientID	Integer	Client Records
Plasterer	Firstname	String/text	Client Records
Plasterer	Surname	String/text	Client Records
Plasterer	AddrLine1	String/text	Client Records
Plasterer	AddrLine2	String/text	Client Records
Plasterer	AddrLine3	String/text	Client Records
Plasterer	AddrLine4	String/text	Client Records
Plasterer	PostCode	String/text	Client Records
Plasterer	Email	String/text	Client Records
Plasterer	MobNumber	String/text	Client Records
Job site	JobID	Integer	Job Records
Job site	ClientID	Integer	Job Records
Job site	Job Desc	Text	Job Records
Job site	AddrLine1	String/text	Job Records
Job site	AddrLine2	String/text	Job Records
Job site	AddrLine3	String/text	Job Records
Job site	AddrLine4	String/text	Job Records
Job site	PostCode	String/text	Job Records
Job site	Job Total Price	Currency	Job Records
Job site	JobPaid	Boolean	Job Records
Job site	JobDaysWorked	Integer	Job Records
Job site	JobComplete	Boolean	Job Records
Plasterer	AppointmentID	Integer	Appointment Records
Plasterer	ClientID	String/text	Appointment Records
Plasterer	PlastererID	String/text	Appointment Records
Plasterer	AppointmentDate	String/text	Appointment Records

Plasterer	AppointmentTime	String/text	Appointment Records
Plasterer	AppointmentAddr1	String/text	Appointment Records
Plasterer	AppointmentAddr2	String/text	Appointment Records
Plasterer	AppointmentAddr3	String/text	Appointment Records
Plasterer	AppointmentAddr4	String/text	Appointment Records
Plasterer	InvoiceID	Integer	Invoice Records
Plasterer	ClientID	Integer	Invoice Records
Plasterer	JobID	Integer	Invoice Records
Plasterer	PlastererID	Integer	Invoice Records
Plasterer	InvoiceAmountPreTax	Currency	Invoice Records
Plasterer	InvoiceAmountAfterTax	Currency	Invoice Records
Plasterer	InvoiceReceived	Boolean	Invoice Records
Plasterer	InvoiceDate	DateTime	Invoice Records
Plasterer	InvoiceText	Text	Invoice Records
Plasterer	JobMaterialsID	Integer	JobMaterials Records
Plasterer	JobID	Integer	JobMaterials Records
Plasterer	MaterialsID	Integer	JobMaterials Records
Plasterer	JobMaterialsQuantity	Integer	JobMaterials Records

Data flow diagrams
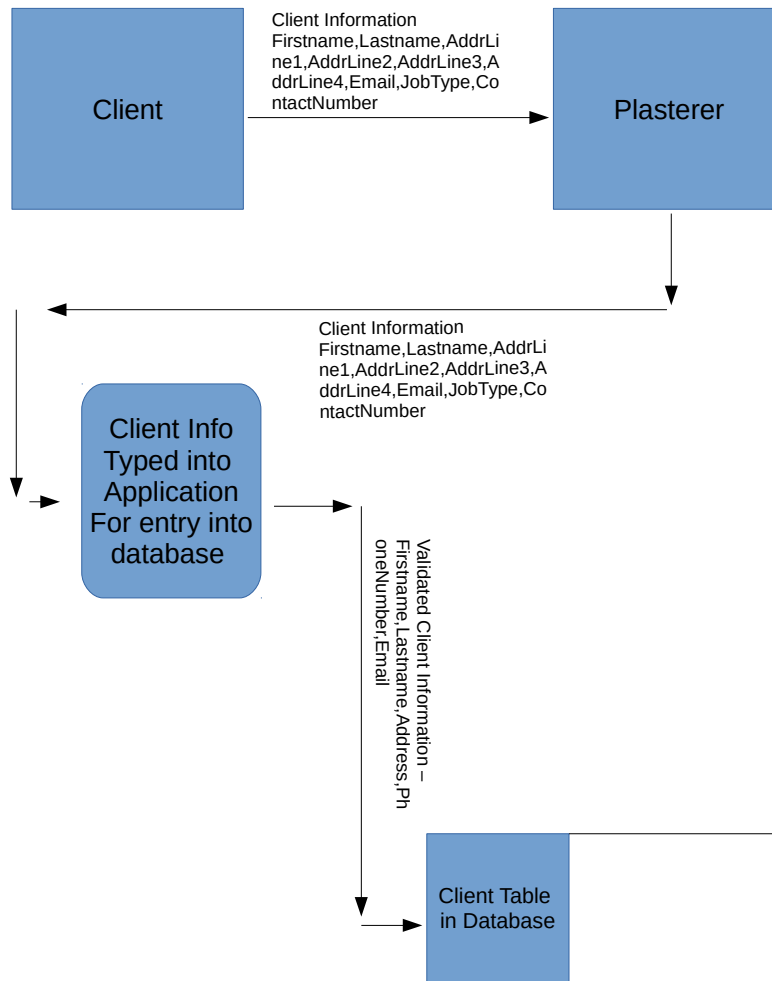
Figure 1.12: This data flow diagram signifies the flow of data in the proposed system when gaining a new clients info.
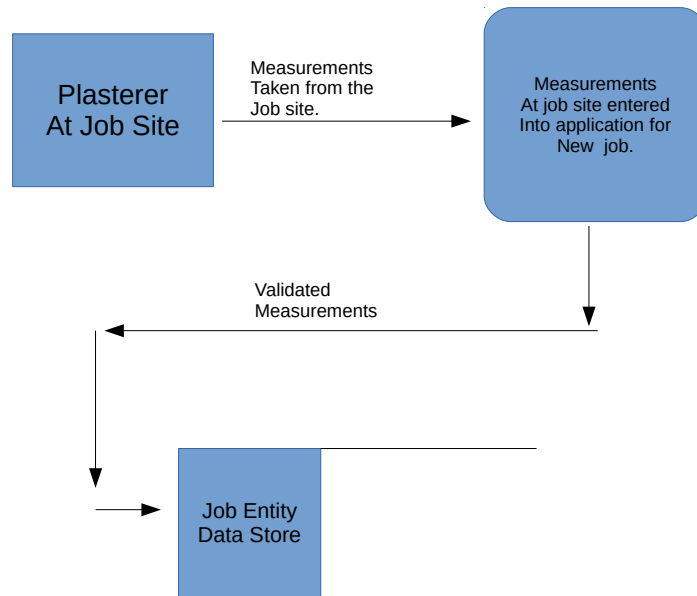
Figure 1.13: This data flow diagram signifies the flow of data in the proposed system when collecting the measurements from the job site.
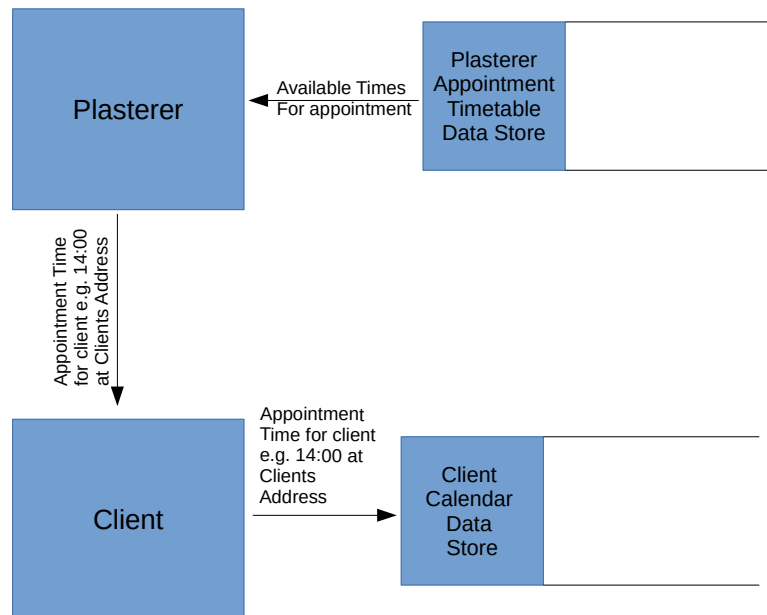
Figure 1.14: This diagram shows the flow of the data in the proposed system when a client is given an appointment time.
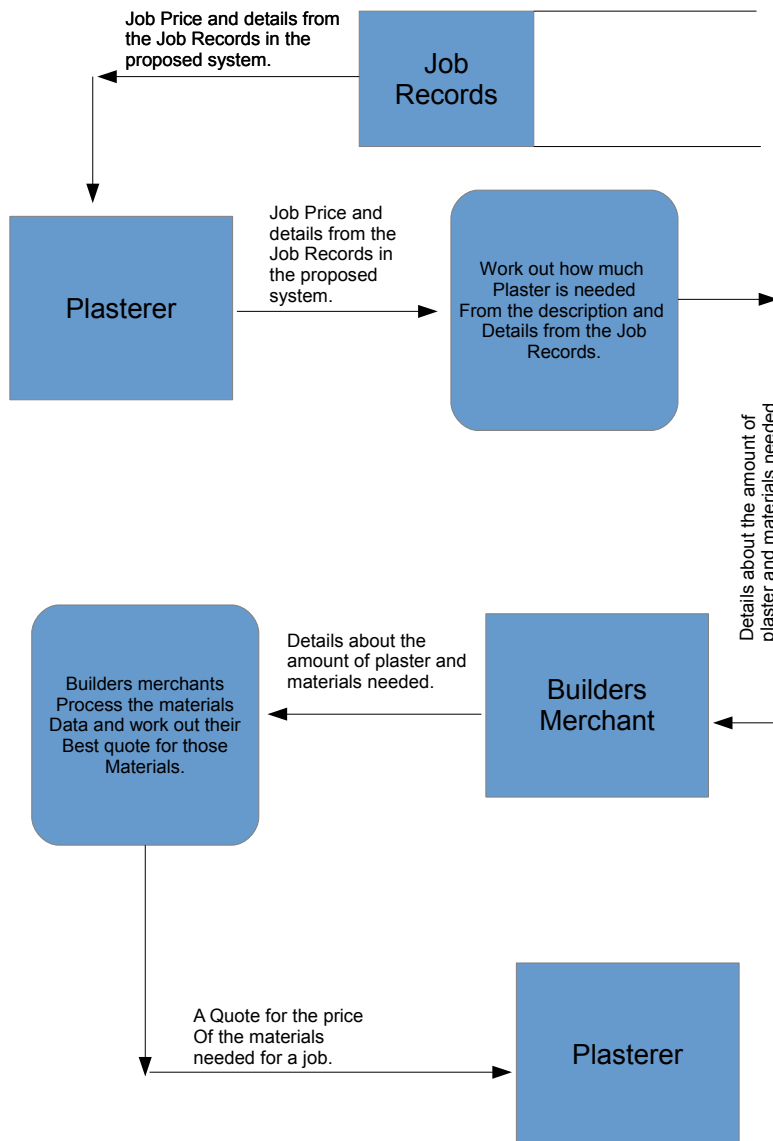
Figure 1.15: This diagram shows the flow of the data in the proposed system when the plasterer gets a quote for the materials from the builders merchants.
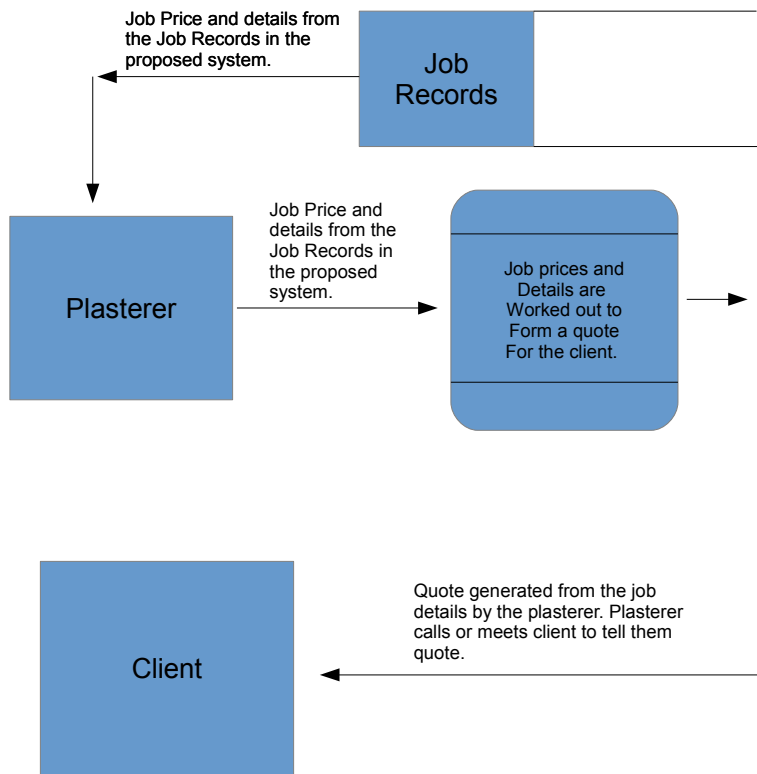
Figure 1.16: This diagram shows the flow of the data in the proposed system when a quote is generated for the client and given to them.

Data dictionary

Name	Data Type	Length	Validation	Example Data	Comment
ClientID	Integer	1 - 1000	Range	1, 200, 45	Primary Key
ClientTitle	String	2 - 10 Chars	Length	Mr,Mrs,Sir	
ClientFirstName	String	3 - 25 Chars	Length	Dan, kyle	
ClientSurname	String	3 - 35 Chars	Length	Austin, Kirkby	
ClientAddrLine1	String	5 - 30 Chars	Length	15 Crowley Road	Street name
ClientAddrLine2	String	5 - 30 Chars	Length	Haverhill	Town / City
ClientAddrLine3	String	5 - 30 Chars	Length	Suffolk	County
ClientAddrLine4	String	6 - 7 Chars	Length and Format Check	CB9 0DJ	Post Code
ClientEmail	String	7 - 50 Chars	Length and format check	dan@ gmail.com	
ClientPhoneNumber	String	11 Chars	Length	07809 726 812	
PlastererID	Integer	1 - 50	Range	1, 2, 45	Primary Key
PlastererFirstName	String	3 - 30 Chars	Length	Dan, Kyle	
PlastererSurname	String	3 - 30 Chars	Length	Austin, Kirkby	
PlastererAddrLine1	String	5 - 30 Chars	Length	15 Long Road	Street Name
PlastererAddrLine2	String	5 - 30 Chars	Length	Cambridge	Town / City
PlastererAddrLine3	String	5 - 30 Chars	Length	Essex	County
PlastererAddrLine4	String	6 - 7 Chars	Length and Format Check	CB2 5HD	Post Code
PlastererEmail	String	7 - 50 Chars	Length and Format Check	dan@ gmail.com	
PlastererPhoneNumber	String	11 Chars	Length	07710 300 678	
PlastererDailyRate	Currency	40 - 250	Range Check	70, 150, 200	
JobID	Integer	1 - 2000	Range Check	1, 3, 5	Primary Key
JobDescription	Text	10 - 1000 Chars	Range Check	5m x 4m x 3m Living Room to be plastered.	Description of job.
JobAddrLine1	String	5 - 30 Chars	Length	15 Long Road	Street Name

JobAddrLine2	String	5 - 30 Chars	Length	Cambridge	Town / City
JobAddrLine3	String	5 - 30 Chars	Length	Essex	County
JobAddrLine4	String	6 - 7 Chars	Length and Format Check	CB2 5HD	Post Code
JobDaysWorked	Integer	0 - 30	Range Check	1, 7 14	
JobComplete	Boolean		Presence Check	TRUE, FALSE	
JobPaid	Boolean		Presence Check	TRUE, FALSE	
MaterialID	Integer	1 - 50	Range Check	3, 4, 29	Primary Key
MaterialName	String	3 - 50	Length	Angle Beading	
MaterialPrice	Currency	1 - 1000	Range Check	1, 3, 20	
JobMaterialsID	Integer	1 - 4000	Range Check	1, 300, 3563	Primary Key
JobMaterialsQuantity	Integer	1 - 500	Range Check	59, 245, 309	
InvoiceID	Integer	1 - 2000	Range Check	1, 2, 45	Primary Key
InvoiceAmountPreTax	Currency	50 - 7000	Range Check	140, 890, 1050	
InvoiceAmountAfterTax	Currency	50 - 7000	Range Check	100, 800, 900	
InvoiceReceived	Boolean		Presence Check	TRUE, FALSE	
InvoiceDate	DateTime		Presence Check	14/12/2013 14:20	
InvoiceText	Text	1 - 1000	Length	5 Days Worked	
AppointmentID	Integer	1 - 2500	Range Check	1, 2, 45	Primary Key
AppointmentDate	Date		Presence Check	07/12/2013	
AppointmentTime	Time		Presence Check	16:10	
AppointmentAddrLine1	String	5 - 30 Chars	Length	15 Long Road	Street Name
AppointmentAddrLine2	String	5 - 30 Chars	Length	Cambridge	Town / City
AppointmentAddrLine3	String	5 - 30 Chars	Length	Essex	County
AppointmentAddrLine4	String	6 - 7 Chars	Length and Format Check	CB2 5HD	Post Code

Volumetrics

In the current system Dan only has a few regular clients but it is possible to receive up to 4-5 additional clients each week (mentioned in interview question 4 6). Therefore using the guidline of around 4 clients a week and a trial period for the application of 3 months, there could be upto 48 new clients (4 clients a week x 4 weeks in month x 3 months = 48) added into the database. It would also be useful to add the existing clients from paper so another 50 clients could be added from what is already stored manually. So around 100 clients may be added to the new system. This size can be increased later if necessary.

Below I have established the various attribute types and the relative storage space required to store them.

Attribute sizes for proposed SQL database in project:

- Boolean - 1 Byte (Stored as 0 or 1 in sqlite3)

- Integer - 1 - 8 Bytes depending on size of number (we will use the middle value for calculations - 4)

- Real - 8 Byte Floating point number

- Text - Depends on size of text (1 Byte per character)

In the interview (6) Dan said that only a few lines of text are put in the workbook each time data is entered. A few lines may be roughly 50 Characters on average, give or take a few characters. So in the calculations below Text attributes will be 50 Bytes, Boolean attributes will be 1 Byte, Integer attributes will be 4 Bytes(average) and Real/Currency attributes will be 8 Bytes.

- Integers = 9 Attributes

- Booleans = 3 Attributes

- Real/Currency = 4 Attributes

- String/Text = 31 Attributes (Including date and time)

From the above values I can calculate an average attribute/field size to use in calculations:

- Total number of attributes = 47

- Calculating the average attribute size:

- = (9 * 4 B) + (3 * 1B) + (4 * 8 B) + (31 * 30B)

- = 1001

- = 1001 / 47

- = 21.3 Bytes

In the job details table each client may have up to 10 or more jobs. Some clients will only have one and some will be recurring customers so will have a few more. All together there are 37 attributes (minimum) per client that will need to be added and some clients have more than one job with Dan. So for each client lets say there are 100 fields at 21.3 Bytes each added to the database. Below are the calculations to find out the required storage space for the proposed system.

- 100 x 21.3 B = 2130 Bytes

- 2130 / 1024 = 2 KB

- 2KB * 100 = 200 KB

- 200 KB + 200KB(Additional fields and database structure) = 400KB

If the application took up around 10MB itself then the total space required for the proposed system would be around (10MB + 0.4MB = 10.4MB). Dans computer has plenty of hard disk space that could be used when installing the application. This proposed system would therefore have enough storage space for 100 clients to be added to the client database.

1.3   Objectives

1.3.1   General Objectives

- Clean and easy to use GUI.

- Use a database for storing data.

- Make it as easy as possible to find data.

- Add clients to database.

- Make it easy to calculate costs involved.

- Sort client data and add search functionality.

- Ability to add multiple jobs per client.

1.3.2   Specific Objectives

Client Data Store Objectives

- Ability for Dan to add a client to a database.

- Ability for Dan to be able to delete a client from the database if needed.

- Dan should be able to modify and append exisitng clients details with and easy to use system.

- A search feature that will let Dan filter the database of clients to dfind vital information.

Jobs Objectives

- Dan will be able to add a job which will relate to a client stored in the client entity.

- Multiple jobs can be added.

- The jobs will contain the job details (description) and address etc (see entity descriptions for more details).

- Each job will be able to generate invoices from the data.

- The invoice will be able to be sent to the client digitally(by email) or manually (print invoice).

- The user will have the ability to edit the details if needed for each job.

Management features objectives.

- Dan will be able to generate reports between x and y time periods in order to see the amount earned within that period.

- A feature will be implemented whereby the user can deduct tax and other specified costs from the amount earned within that period.

- Ability to print this pay report and generate a digital versatile copy of the report.

1.3.3   Core Objectives

- The application must store the client details in a database.

- The application must be able to add jobs for each client.

- The application must be able modify client and job details.

- The application must be able to send an invoice to the client via email.

- The application must be able to generate a report for the amount earned within a time period.

1.3.4   Other Objectives

- The application may be able to print an invoice for the client.

- The application may be able to send quotes for jobs via email.

- The application may be able to print a report for the amount earned within a time period.

1.4   ER Diagrams and Descriptions

1.4.1   ER Diagram
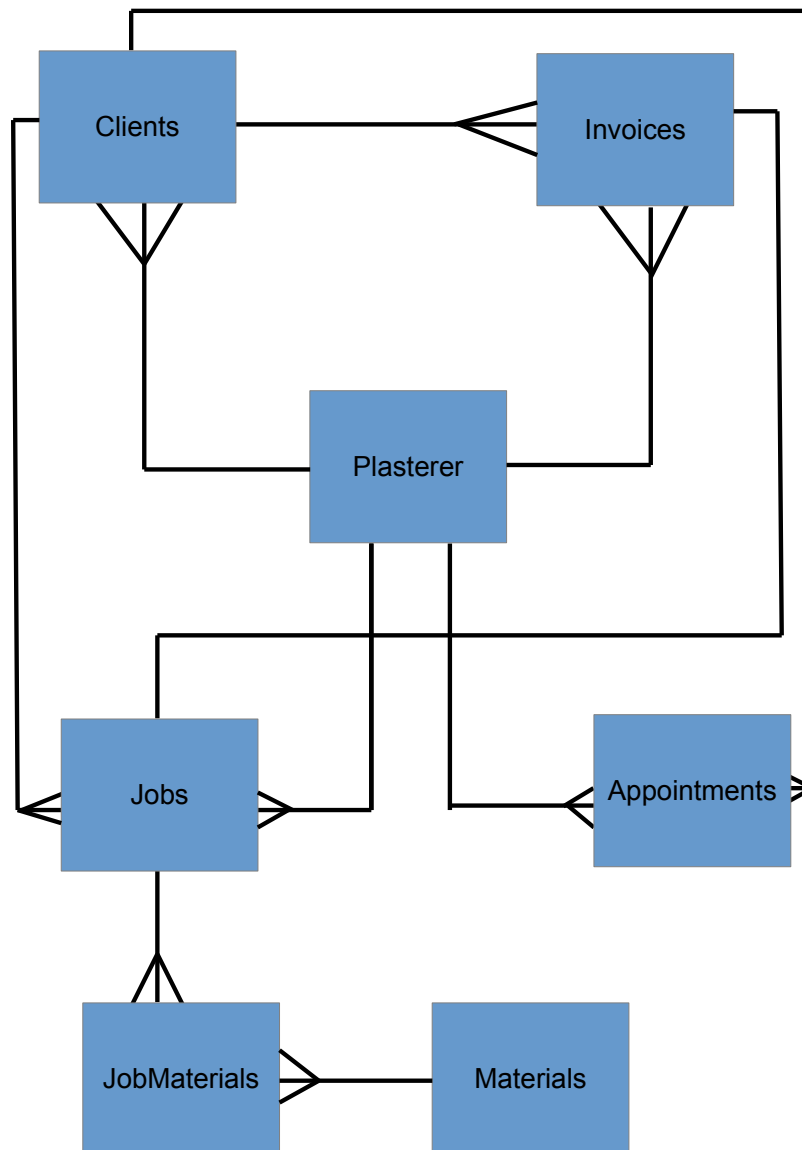
Figure 1.17: This is the entity relationship diagram for the sqlite3 database.

1.4.2 Entity Descriptions

Below are the entity descriptions for the various entites in the proposed system. An underlined attribute denotes a primary key and a *emphasised* attribute signifies a foreign key in the entity.

Client(ClientID, ClientTitle, ClientFirstName, ClientSurname, ClientAddrLine1, ClientAddrLine2, ClientAddrLine3, ClientAddrLine4, ClientEmail, ClientPhoneNumber, *PlastererID*)

Plasterer(PlastererID, PlastererFirstName, PlastererSurname, PlastererAddrLine1, PlastererAddrLine2, PlastererAddrLine3, PlastererAddrLine4, PlastererEmail,PlastererPhoneNumber, PlastererDailyRate)

Job(JobID, *ClientID*, *PlastererID*, JobDescription, JobAddrLine1, JobAddrLine2, JobAddrLine3, JobAddrLine4, JobDaysWorked, JobComplete, JobPaid, *InvoiceID*)

Material(MaterialID,MaterialName,MaterialPrice)

JobMaterials(JobMaterialsID, *JobID*, *MaterialsID*, JobMaterialsQuantity)

Invoice(InvoiceID, *ClientID*, *JobID*, *PlastererID* InvoiceAmountPreTax, InvoiceAmountAfterTax, InvoiceReceived, InvoiceDate, InvoiceText)

Appointment(AppointmentID, *ClientID*, *PlastererID*, AppointmentDate, AppointmentTime, AppointmentAddrLine1, AppointmentAddrLine2, AppointmentAddrLine3, AppointmentAddrLine4)

1.5   Object Analysis

1.5.1   Object Listing

Objects

- Client
- Jobs
- Materials
- Invoices
- Appointments
- Plasterer

1.5.2   Relationship diagrams
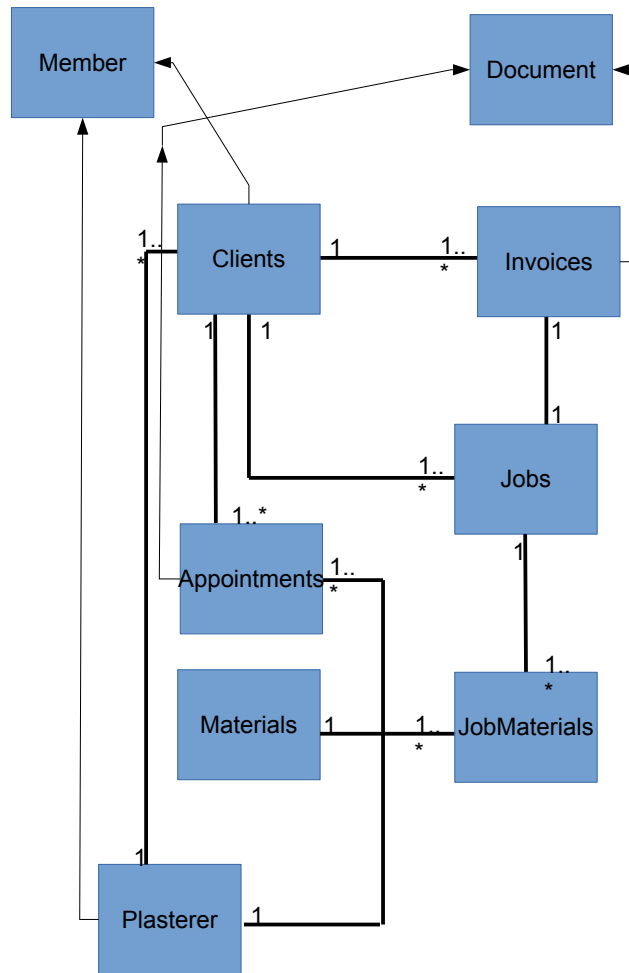
Figure 1.18: Relationship Diagram for the entities in the proposed system.

1.5.3   Class definitions

Key:

Label
Attributes
Behavious

Member
MemberID
MemberTitle
MembeFirstName
MemberSurname
MemberAddrLine1
MemberAddrLine2
MemberAddrLine3
MemberAddrLine4
MemberEmail
MemberPhoneNumber
AddMemberTitle
AddMemberFirstName
AddMemberSurname
AddMemberAddrLine1
AddMemberAddrLine2
AddMemberAddrLine3
AddMemberAddrLine4
AddMemberEmail
AddMemberPhoneNumber
EditMemberTitle
EditMemberFirstName
EditMemberSurname
EditMemberAddrLine1
EditMemberAddrLine2
EditMemberAddrLine3
EditMemberAddrLine4
EditMemberEmail
EditMemberPhoneNumber

Client: extends **Member**

Plasterer: extends **Member**
PlastererDailyRate
AddPlastererDailyRate
EditPlastererDailyRate

Job:
JobID
ClientID
PlastererID
JobDescription
JobAddrLine1
JobAddrLine2
JobAddrLine3
JobAddrLine4
JobDaysWorked
JobComplete
JobPaid
InvoiceID
AddJobDescription
AddJobAddrLine1
AddJobAddrLine2
AddJobAddrLine3
AddJobAddrLine4
AddJobDaysWorked
EditJobComplete
EditJobPaid
EditJobDescription
EditJobAddrLine1
EditJobAddrLine2
EditJobAddrLine3
EditJobAddrLine4
EditJobDaysWorked

Material:
MaterialID
MaterialName
MaterialPrice
AddMaterialName
AddMaterialPrice
EditMaterialName
EditMaterialPrice

Document:
DocumentID
DocumentDate
DocumentTime
DocumentText
AddDocumentDate
AddDocumentTime
AddDocumentText
EditDocumentDate
EditDocumentTime
EditDocumentText

Invoice: extends **Document**
ClientID
JobID
PlastererID
InvoiceAmountPreTax
InvoiceAmountAfterTax
InvoiceReceived
AddInvoiceAmountPreTax
AddInvoiceAmountAfterTax
AddInvoiceReceived
EditInvoiceAmountPreTax
EditInvoiceAmountAfterTax
EditInvoiceReceived

Appointment: extends **Document**
ClientID
PlastererID
AppointmentAddrLine1
AppointmentAddrLine2
AppointmentAddrLine3
AppointmentAddrLine4
AddAppointmentAddrLine1
AddAppointmentAddrLine2
AddAppointmentAddrLine3
AddAppointmentAddrLine4
EditAppointmentAddrLine1
EditAppointmentAddrLine2
EditAppointmentAddrLine3
EditAppointmentAddrLine4

1.6   Other Abstractions and Graphs

Due to the nature of this project there are no graphs that could represent the current or proposed systems.

1.7   Constraints

1.7.1   Hardware

Dan currently has a Toshiba Laptop with the following hardware components:

- 6GB DDR3 RAM

- Intel 2.0ghz Dual Core Processor

- High Resolution 1920 x 1080 Display

- 1 TB HDD

- Intel Onboard Integrated Graphics

The specification of Dan's laptop is more than powerful enough and has enough RAM to run the proposed python application alongside multiple other programs that he regularly uses (such as a Web Browser and Media Player). The hard drive has enough storage space to install the application (which will only be around 10MB) so storage is not a problem. The only possible complication when it comes to hardware may be the screen resolution as the application will have to keep within the resolution and be optimized for this screen size and many more to keep the software versatile.

1.7.2   Software

Dan is running the Windows 8 operating system on his laptop which does not have Python 3 installed. The proposed system will need to be able to run on this operating system therefore it is a software constraint that will need to be addressed. This could be achieved by building an installer for the python application so it can be installed on Windows 8 as pythons libraries are included.

1.7.3   Time

Dan is very flexible when it comes to time as he is a freelancer and is in no rush for a replacement to his old system. He does not mind how long the new system takes to be implemented just that it works and functions as expected.

Therefore the only time constraint for the proposed system would be the implementation deadline which is Friday 13th February 2015.

1.7.4 User Knowledge

Dan has no formal qualifications in IT and has never performed complex computerised tasks. Browsing the internet, watching films and playing music is the extent to which Dan currently uses his laptop. Therefore user knowledge is a possible constraint and the proposed system will need to be easy to navigate and use. This may be achieved by providing a simple non-complex GUI and detailed tutorials/documentation. Also any errors must be handled correctly as Dan does not know Python and the erros would be useless in that format to him. This can be handled with user friendly exceptions put in throughout the program.

1.7.5 Access restrictions

The proposed system does not have any specific requirements when it comes to security however it needs to conform to the Data Protection Act 1998 which requires personal information to be stored securely, accurately and up to date. This is a possible constraint when it comes to access restrictions and will need to be addressed in order to not breach the UK legislation on this. The database may need to also be encrypted to not reveal personal information in plain text to any malicious threats.

1.8 Limitations

1.8.1 Areas which will not be included in computerisation

- The collection of the measurements, which is currently being done manually, may be implemented using a device which has kinect like sensors, you could scan a room and pick up the measurements of it using just a phone for example. This might be possible.

1.8.2 Areas considered for future computerisation

- Getting the quote for the price of the materials will still be done manually as it will be difficult to computerise as you would have to visit the Builders Merchants in person. But in the future an API might be available from selected builders merchants to get live quotes for materials.

1.9   Solutions

1.9.1   Alternative solutions

Solution	Advantages	Disadvantages
Web Application	Can be accessed on multiple devices where theres is an internet connection. Only a web browser required (already installed in most cases). Can be easier to make it look "pretty" as there are many graphical libraries for the web. Can use different server side languages to program the backend of the website. (Python (Django),Ruby(RoR) ,PHP(Zend,Magneto)).	Need an internet connection to visit site if not hosted locally. A web host may need to be purchased. Need to optimize for different browsers (Internet Explorer, Chrome, Firefox, Safari). Problems may happen with host which means the client may not be able to access their application (site may be subject to DDOS (may need to purchase Cloudflare or similiar service)).
Mobile Phone Application	Can mean the application is extremely mobile so the client will be able to take the app wherever he goes which may useful to take to jobs. Client is familiar with using a mobile phone so may be easier for them to use.	Will take longer to complete. It is more complex and can be solved easier with a different method. Client may lose phone which would then result in loss of application.
Command-Line Application	No long waiting time for GUI to load. System resources are not used up as much.	Can be more complicated, so it seems to the client. Interface would be less effective and harder to use for the non-technical client. Errors may occur and look unfamiliar to client. If laptop is lost/stolen the the program may be lost too so backups need to be kept online (not so much a problem with a web application).

Spreadsheet	Can be faster than a application with a GUI to load. Data can be exported easily. Queries are easy to run and execute fast.	No abstraction from the data. Client would have to familiarise themselves with the syntax for queries. You can add a lot more features in PyQt and Python Applications. No GUI to make it easier for the client to control the functions and run queries.
PyQt4 GUI and Python3 Application	Python is easy to write and can therefore get the project done quicker leaving more time for debugging and testing. The GUI will make it a lot easier for the client to use the application. With this form of application it can be packaged for Mac,Windows and Linux Systems. GUI gives a clear visual representation of the applications features.	Uses python which needs to be packaged with the program as windows does not have Python natively installed but Mac and Linux do. Writing a GUI is a lot more work than other solutions.

1.9.2   Justification of chosen solution

I have chosen to use Python 3 alongside the Qt Framework (PyQt4) to create
a standalone application. This is due to the versatility which Python and
PyQt4 offers. Using this method to program a solution to my clients problem I
will be able to get it done effeciently due to my existing knowledge of Python
and I will be able to generate different executables for different operating
systems which may be beneficial to the client as he is looking to buy a Apple
System. It will also offer a visual representation through the use of GUI as
opposed to hard to use Console application approach; which would not be
suitable for the client as he does not have enough technical knowledge with
computers to understand and use it easily - therefore going for something with
a GUI would be a good choice. A web application may have been good but
sometimes Dan takes his laptop where there is no internet connection and
hosting the site locally would not be feasible. Also there are hidden costs
involved with purhcasing a domain and hosting if a web application solution
was chosen. A mobile application may have also been good but would have
taken longer to accomplish due to the time it takes to develop complex mobile
phone applications. A spreadsheet would also not have been the perfect
solution as it requires knowledge of writing queries which, due to Dan's lack of
technical computer knowledge, he does not have. Therefore I believe that a
Python and PyQt4 application would be the best solution to go for.

Chapter 2

Design

2.1 Overall System Design

2.1.1 Short description of the main parts of the system

Main Parts of the System
These are the main parts of the proposed system.

- Proposed System User Interface

- Adding a New Job

- Adding a New Client

- Adding a New Material

- Sorting and Searching Clients

- Removing Clients or Jobs

- Calculating Costs For Each Job

- Generating Reports

- Invoice Output for Client

- Appointment Output for Client

Proposed System User Interface

- Once onto the proposed system, the plasterer will be able to see various buttons in the action bar at the top of the program; these include Add Jobs, Clients, Materials.

- Pressing the Jobs Button will then take them to a different user interface which will then display a series of other options that are applicable under

the Jobs section. These include Add Job, Delete Job, Search Jobs, Edit Job.

Adding a New Job

- The plasterer will click on a + new job button on the main window and will then be shown a different layout allowing the plasterer to add the details of a new job to the database.

- Whilst entering the job details various validation techniques will be implemented on the data entered into the fields. For example, there will be a post code regular expression validation to make sure that the post code entered is one of a correct and valid format.

- Once the form has been validated and submitted the plasterer will be show a success (or failure) message to let them know that it was added ok (or not).

Adding a New Client

- The user will be able to click on a + new client button that will be situated on the main window and once clicked on, the user will be shown a new layout allowing the user to enter a new client and add all of their details to the application database.

- Whilst entering the new client info the data that is entered will be validated and the line edit will change colour depending on whether that data entered is valid or not. For the client details there will also be a regular expression validator attached to various fields including the client phone number, client email and client post code.

- Once the form data has been validated and is ok the data will be committed to the database and the user will be shown a message telling them whether it added the new client to the database successfully or not. They will then be returned to the main layout.

Adding a New Material

- The user will be able to add multiple materials to the database to use when calculating the cost of a job.

- The user will be able to click a + new material button on the main window and then will be shown a new layout allowing them to enter the details and cost etc of the new material. The data entered will also be validated.

- Once the new material is added a success or failure message will be displayed to the user and then they will be returned to the main layout.

Sorting and Searching Clients

- The user will be able to press a "Search Clients" button on the main window and then they will be taken to a new layout showing them a table with a list of the clients and their details.

- Below the table widget their will be a search field that allows the user to search for specific clients quickly and efficiently.

- The results will be updated on the text Changed event.

- The user will be able to sort the clients not only by search but other attributes such as town/city. When the user clicks the sort by town/city push button the results will update and be sorted by town.

Removing Clients and Jobs

- Removing Clients will be a feature available in the clients section of the application. The user will be able to manage the clients in a "Manage Clients" area of the application.

- Their will be a table widget showing a list of all the clients and when a client record is clicked various options will become available to the user such as delete or edit.

Calculating Costs for each Job

- The calculation of the job will occur once the user clicks generate invoice for a specific job.

- This "Generate Invoice" button will be situated within the jobs section of the application (which will be available by clicking the "Jobs" button in the main layout).

- By clicking "Generate Invoice" the algoritm will collaborate all the available information regarding that specific job. See the algorithm section for more detail on this algorithm.

- In addition to this the user (plasterer) will be able to override the resulting calculations if need be before the final details are sent to the client.

Generate Reports

- The reports section will be available by clicking a "Generate Report" button that will in the Jobs section of the application.

- When the generate report button has been clicked a new layout will be displayed showing a table detailing the amount of money earned in a user specified period.

- The time period to generate a report for will be able to be edited through a form below the report table. The form will ask where/when/who to generate a report for.

- Reports will be able to be generated for different plasterers and different clients; or all clients and all plasterers etc.

Invoice Output for Client

- An invoice for a job will be able to be generated by clicking a "Generate Invoice" button on the specific job.

- Once clicked, the invoice will be displayed on screen in a preview box and then when the user is happy with it they can click "Print Invoice" to print the invoice or "Email Invoice" to email the invoice to the client through the email stored for the client within the database.

Appointment Output for Client

- The appointments will be made by clicking the "Setup Appointment" button on the specific job an appointment is needed for.

- When clicked, a new layout will display allowing the user to select an appointment date and time for the client.

- Once validated and inserted into the database, the client will be emailed a copy of the appointment details.

2.1.2 System flowcharts showing an overview of the complete system

Flow Chart Key

This is the key for the following flow charts.
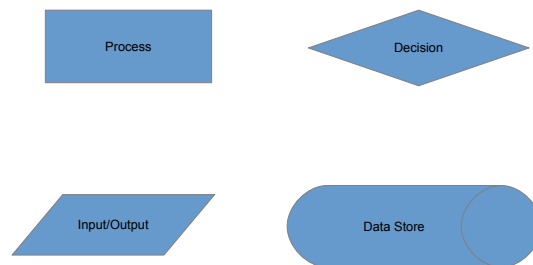
Figure 2.1: This is the flow chart key.

Main Menu Flow Chart

This flow chart shows the options at the main menu in the application.
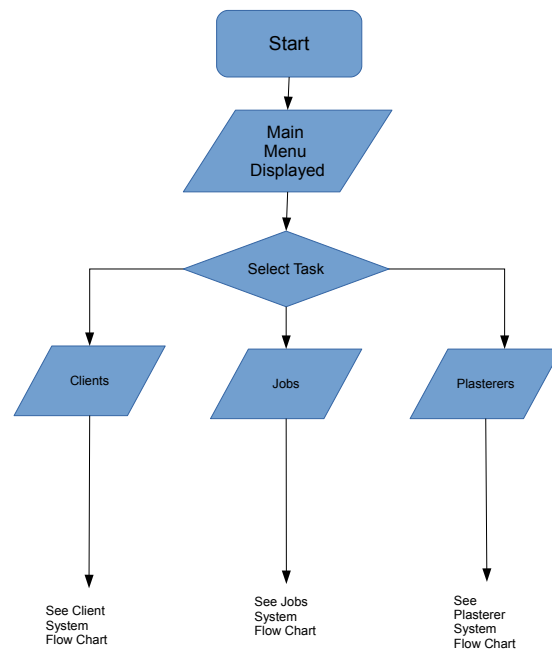
Figure 2.2: This is the flow chart showing the Main Menu Selection.

Client Flow Charts

The flow charts below show the options which can be selected for clients in the application; this includes adding a new client, searching clients and editing clients.

Client Menu Flow Chart

This is the Client Menu flow chart which shows the options to select in the application regarding clients; this includes adding a new client, searching clients and editing clients.
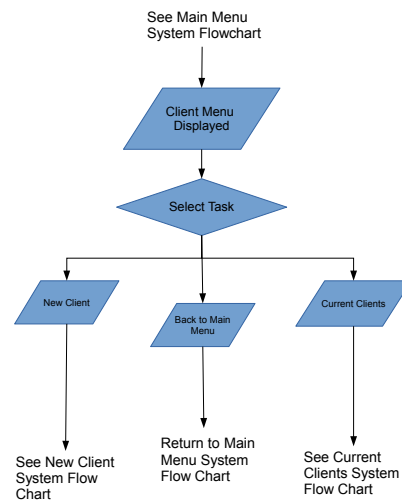
Figure 2.3: This is the flow chart showing the Client Menu Selection.

New Client Flow Chart

Below is a flow chart to show what happens when you add a new client in the proposed system.
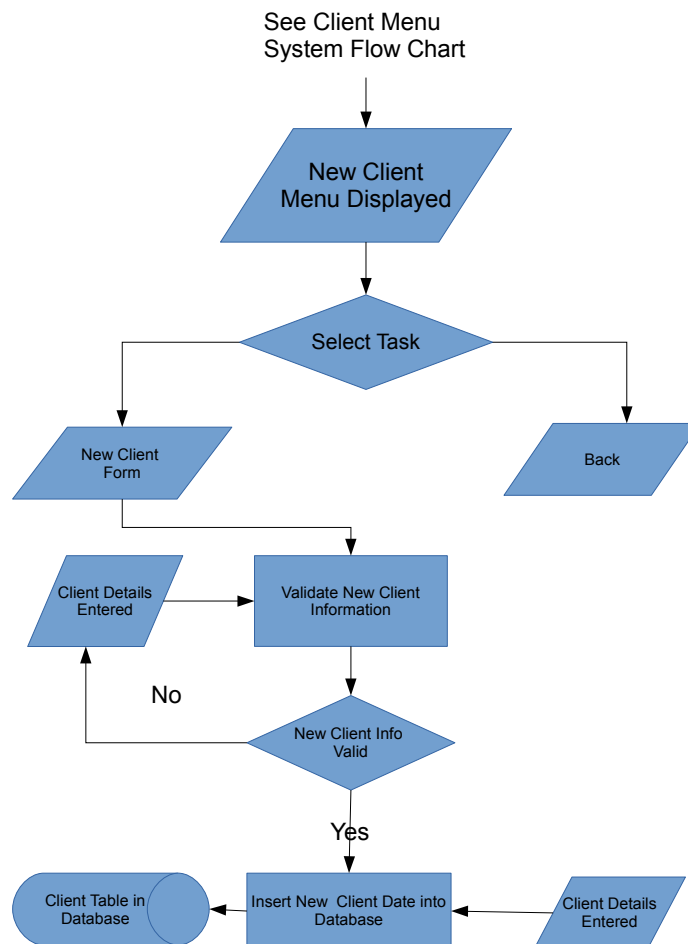
Figure 2.4: This is the flow chart showing the addition of a new client to the proposed system.

Current Clients Flow Chart

Below is the current clients flow chart which shows the flow of control in the proposed system for this section.
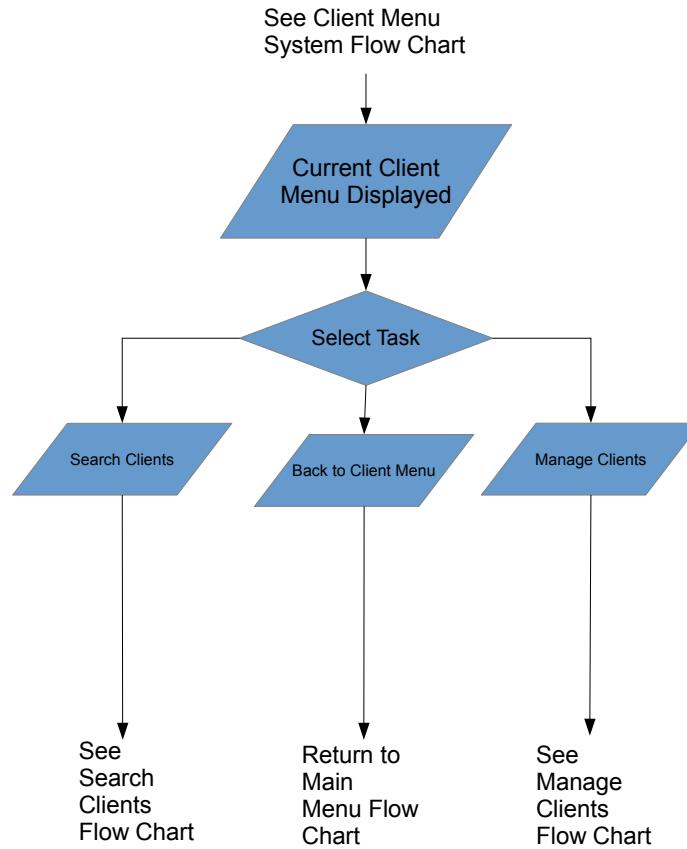
Figure 2.5: This is the Current Clients Flow Chart.

Search Clients Flow Chart

Below you can see how you will be able to search the clients in the proposed system. You will be able to search via Location or Search Term etc.

Return to Current Client Menu Selection

Search Clients Table
Displayed

Sort Clients

Sort By Town/City

Sort By Search Term

Users enters
Search Term

Get Sorted Clients
By Town/City

Get Sorted Clients
By Search Term Entered

Sorted clients by
Tow/city returned
From Database

Sorted clients by
Search Term returned
From Database
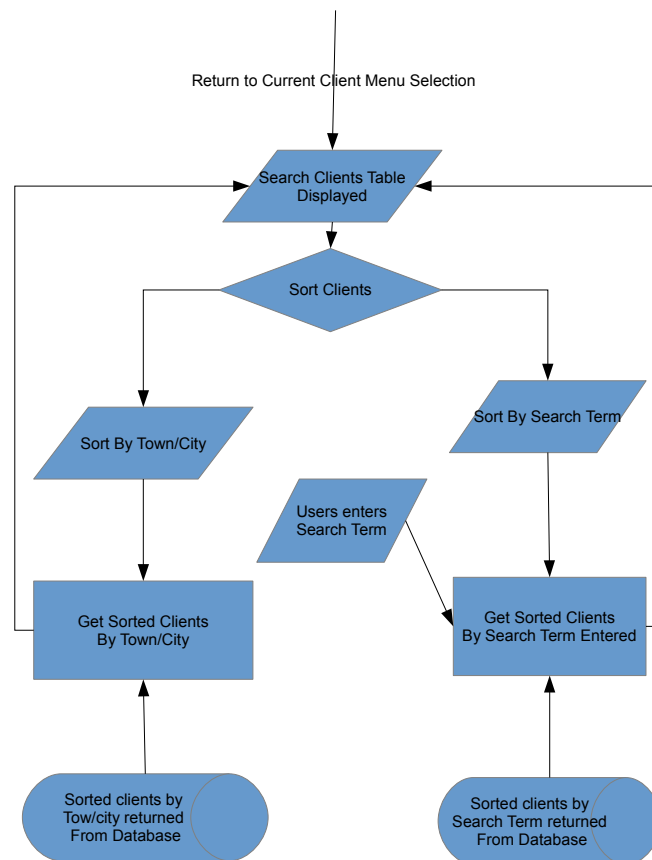
Figure 2.6: This is the Search Clients Flow Chart.

Manage Clients Flow Chart

Below is the manage clients flow chart which shows what happens when you click manage clients in the proposed system.
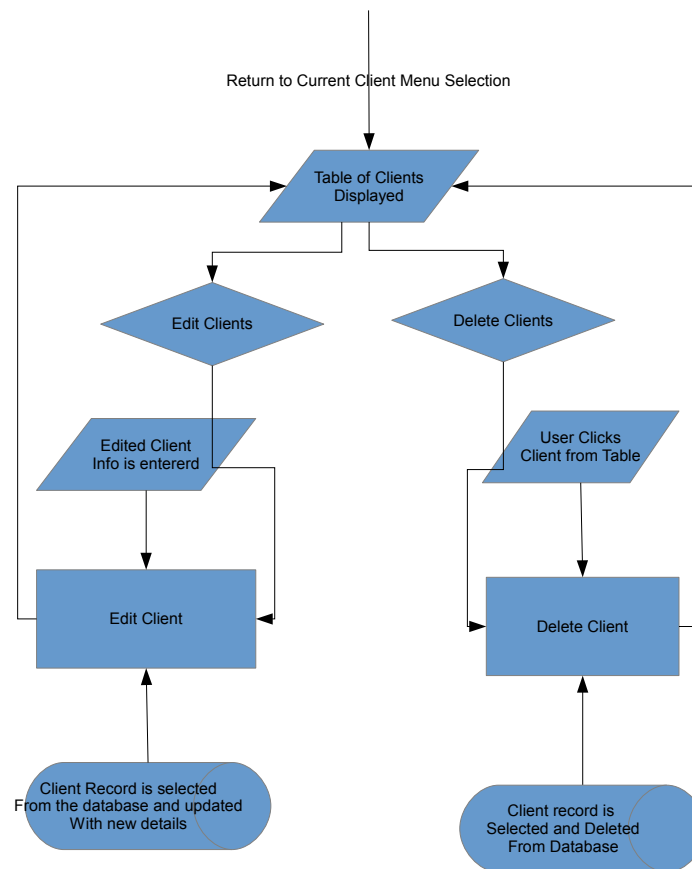
Figure 2.7: This is the Manage Clients Flow Chart.

Plasterer Flow Charts

The flow charts below are from the plasterer section of the application. They show the options that can be selected for plasterers such as searching,editing and creating new plasterers.

Plasterer Menu Flow Chart

This is the plasterer menu flow chart showing what users can do in the plasterers section of the application.
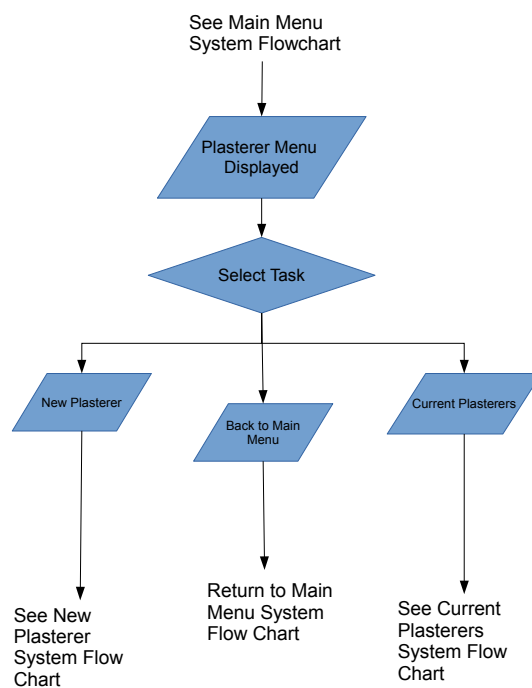
Figure 2.8: This is the Plasterer Menu Flow Chart.

Adding a New Plasterer

This flow chart shows what happens in the application when you add a new plasterer to the system.
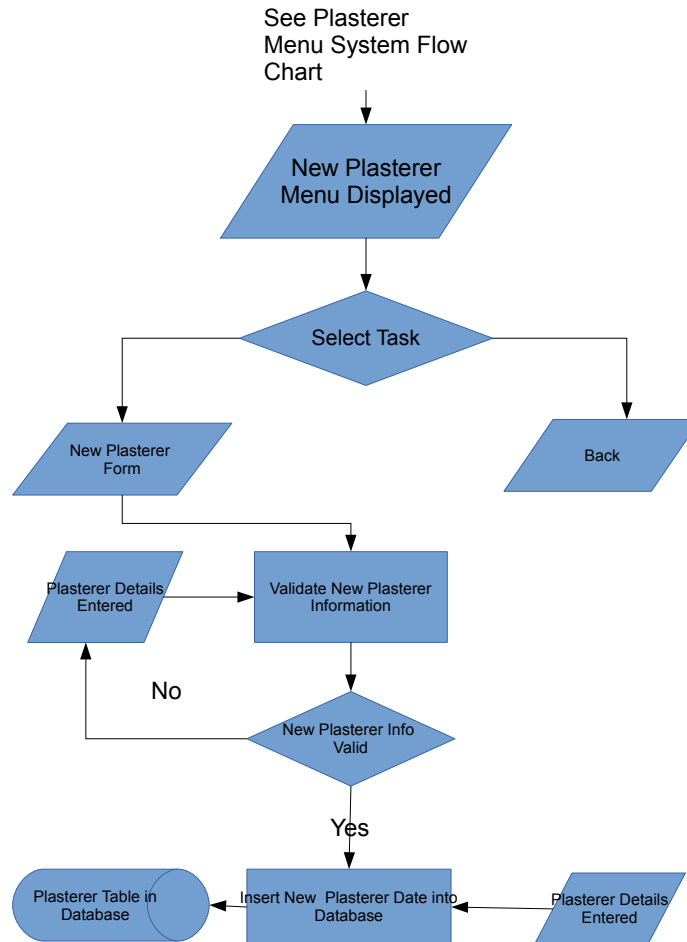
Figure 2.9: This is the New Plasterer Flow Chart.

Current Plasterers

This flow chart shows the current plasterers options from within the application.
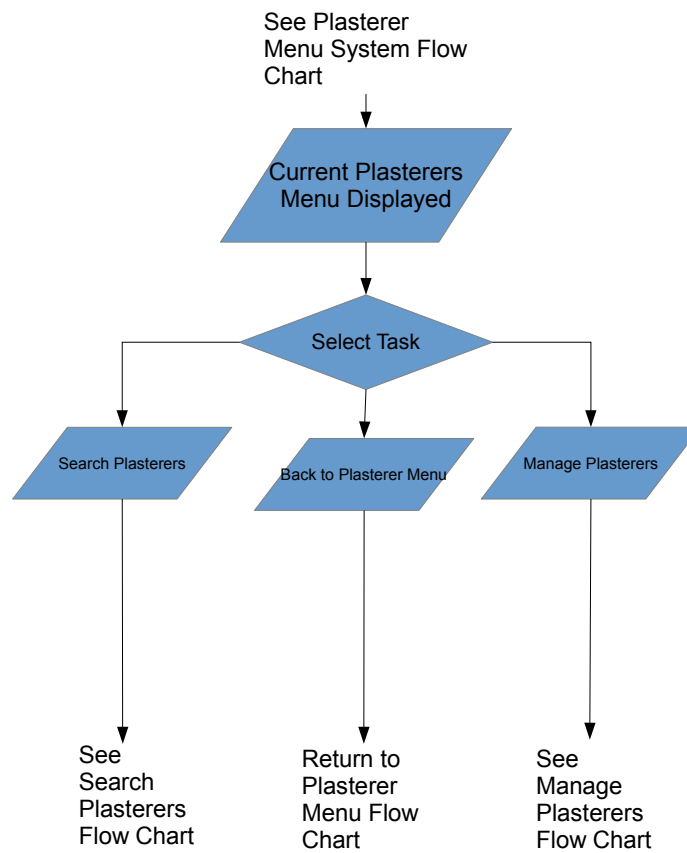
Figure 2.10: This is the Current Plasterers Flow Chart.

Manage Plasterers

This flow chart shows the manage plasterers window from the proposed application.
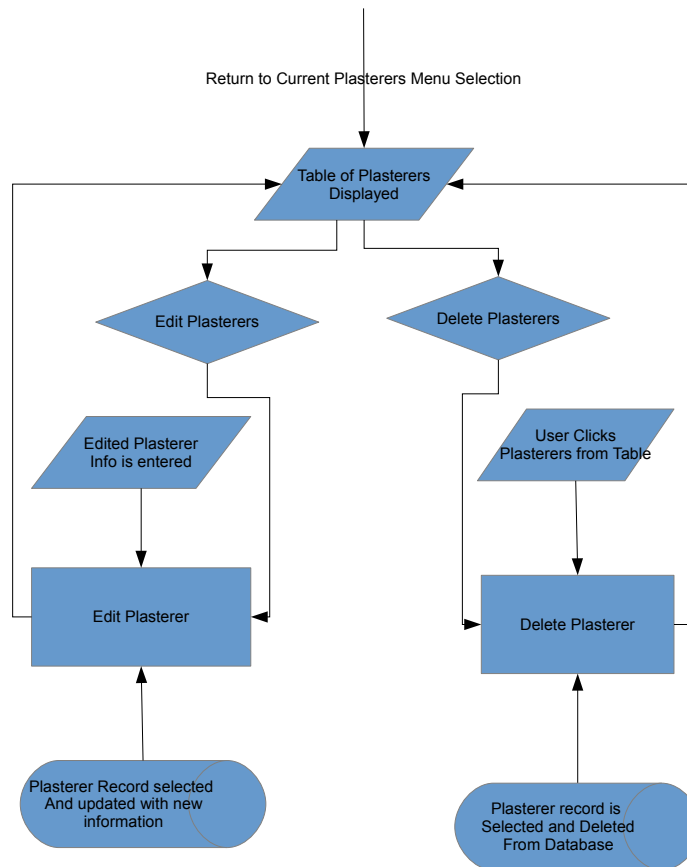
Figure 2.11: This is the Manage Plasterers Flow Chart.

Search Plasterers

This flow chart shows the search plasterers feature of the proposed application. It lets the user search the plasterers for specific records.
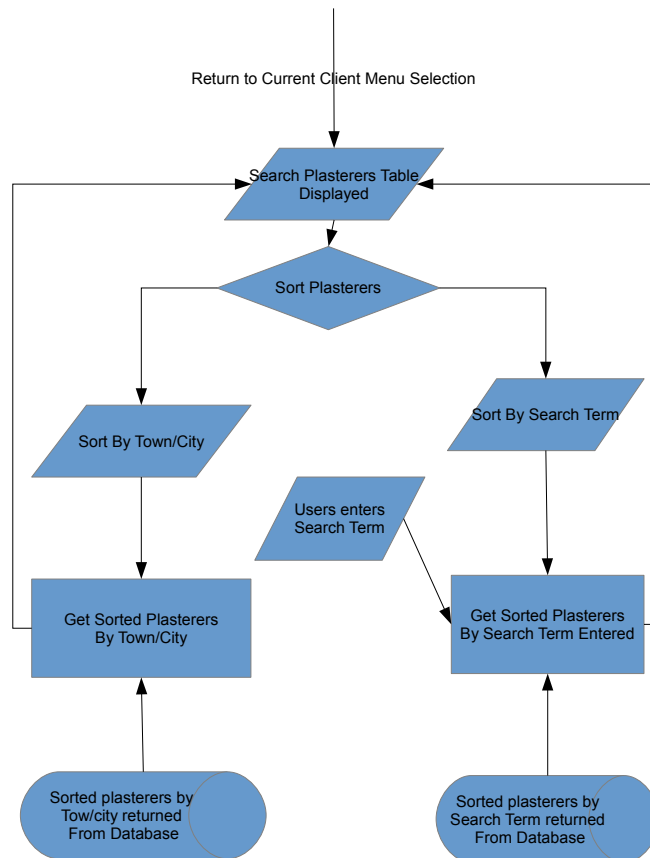
Figure 2.12: This is the Search Plasterers Flow Chart.

Jobs Flow Charts

The flow charts below are from the jobs section of the application.

Job Menu Flow Chart

This flow chart shows the selection choice at the job menu section of the application.
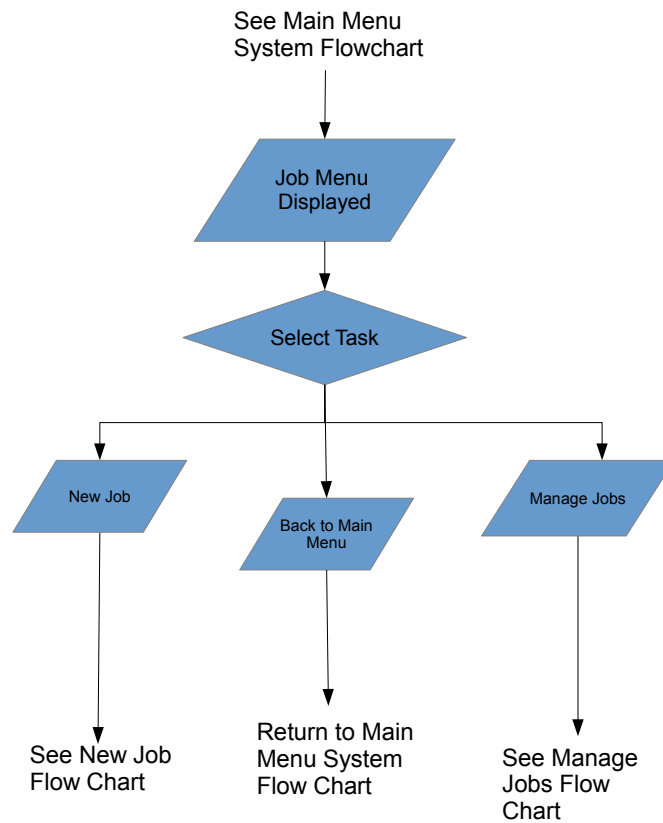
Figure 2.13: This is the Job Menu Flow Chart.

New Job Flow Chart

This flow chart shows what happens in the application when the users adds a new job.
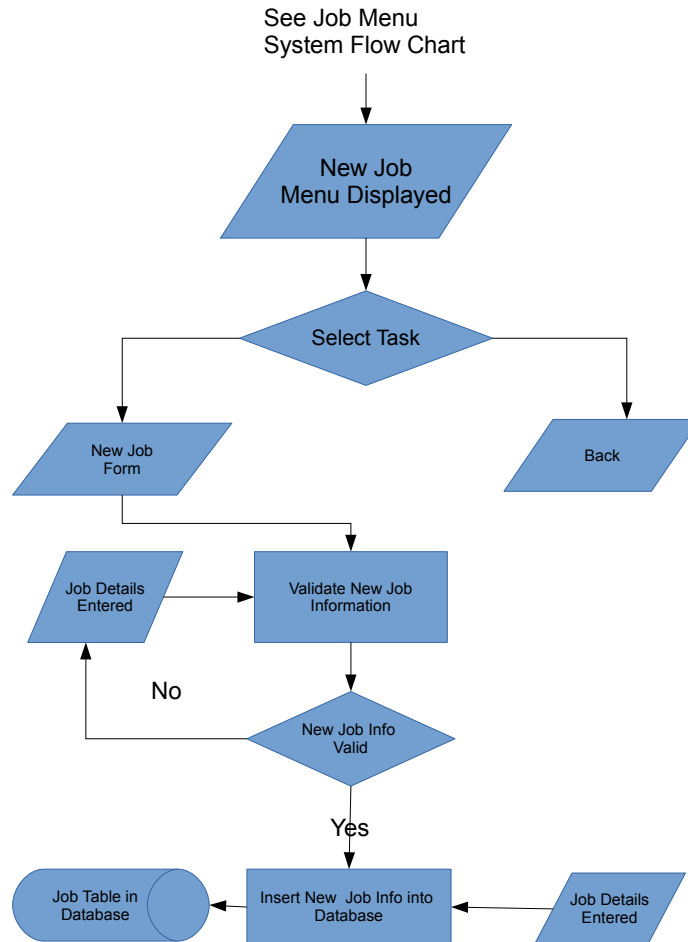
Figure 2.14: This is the New Job Flow Chart.

Manage Jobs Flow Chart

This shows the manage jobs section of the application.
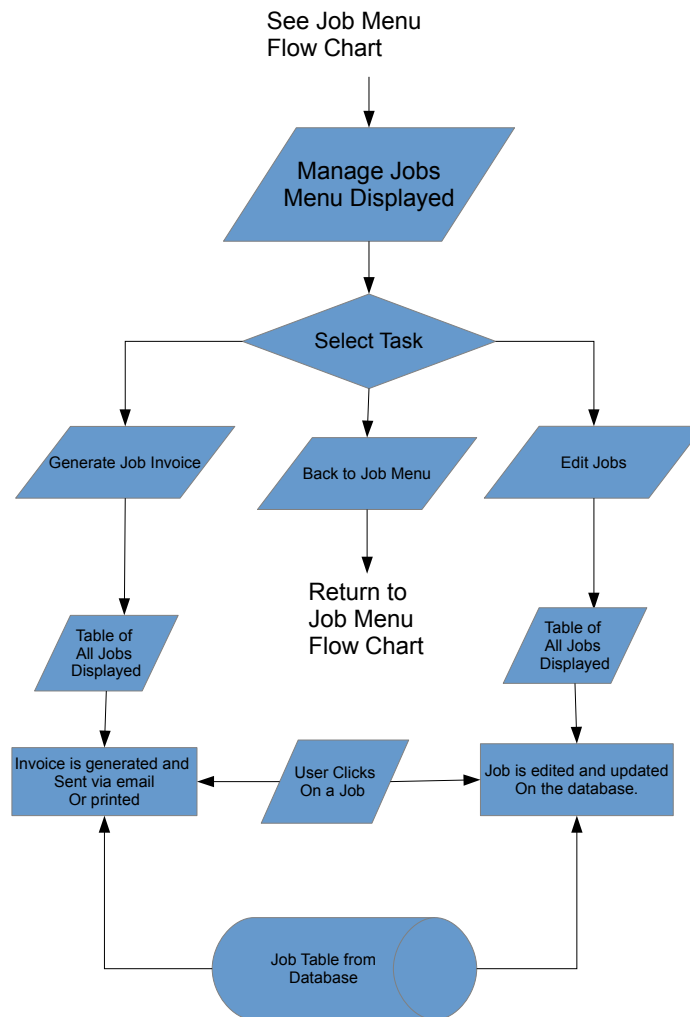
Figure 2.15: This is the Manage Jobs Flow Chart.

2.2   User Interface Designs

Main Menu To Clients Menu UI

This shows what the user interface will look and behave like when the clients button is clicked.
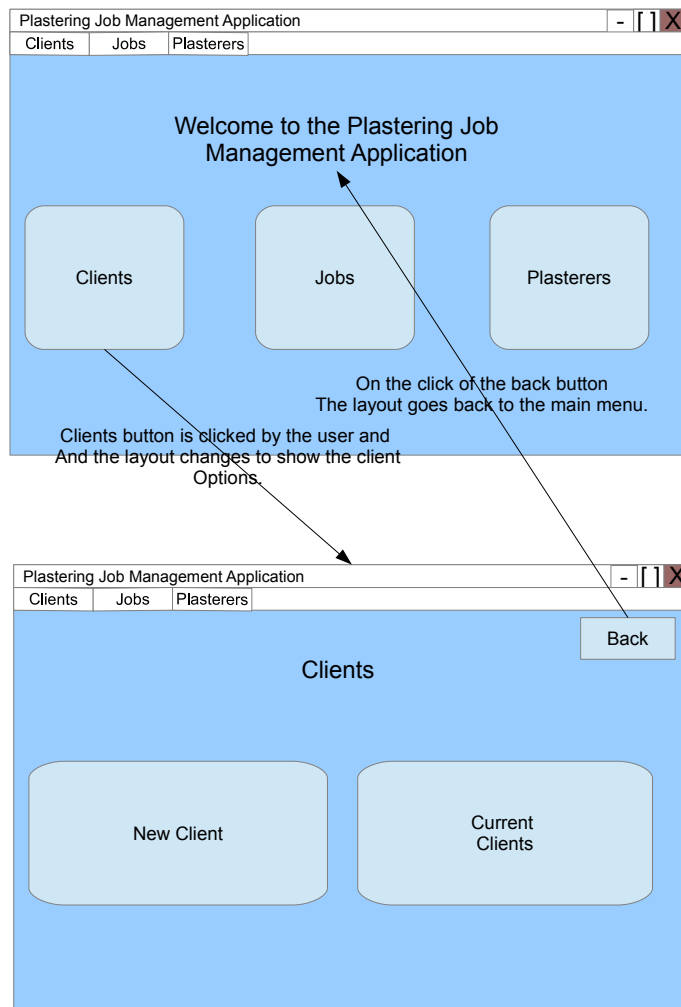
Figure 2.16: This is the Clients Menu User Interface.

Clients Menu to New Client UI

This User Interface diagram shows what happens in the proposed system when
the users adds a new client.
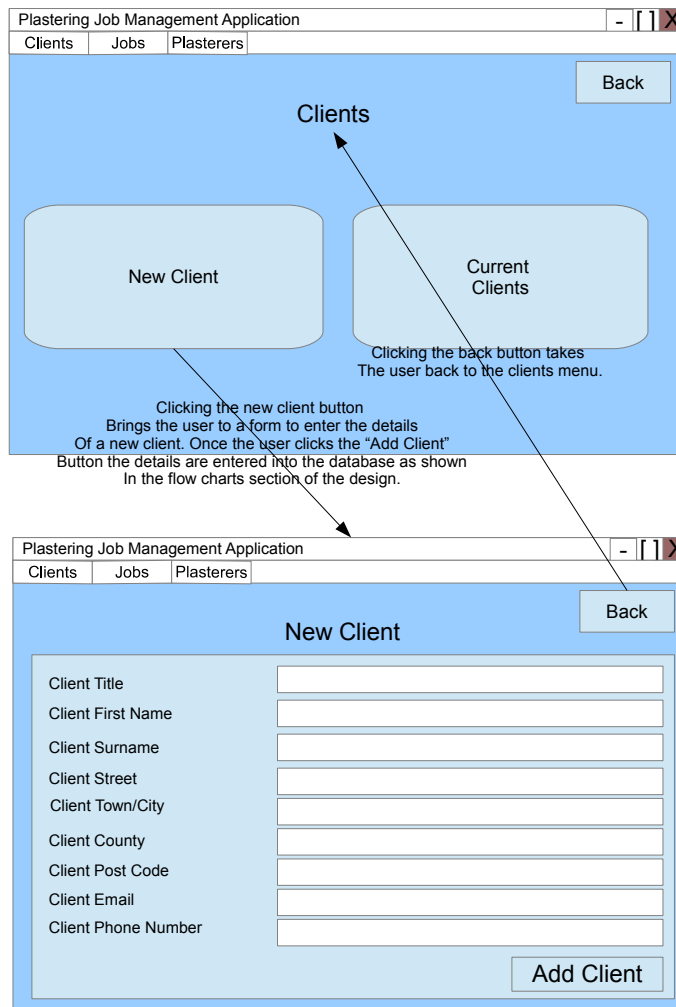
Figure 2.17: This is the New Client User Interface.

2.3   Hardware Specification

2.4   Program Structure

2.4.1   Top-down design structure charts

2.4.2   Algorithms in pseudo-code for each data transformation process

2.4.3   Object Diagrams

This Diagram shows the relationship between the objects in the system.
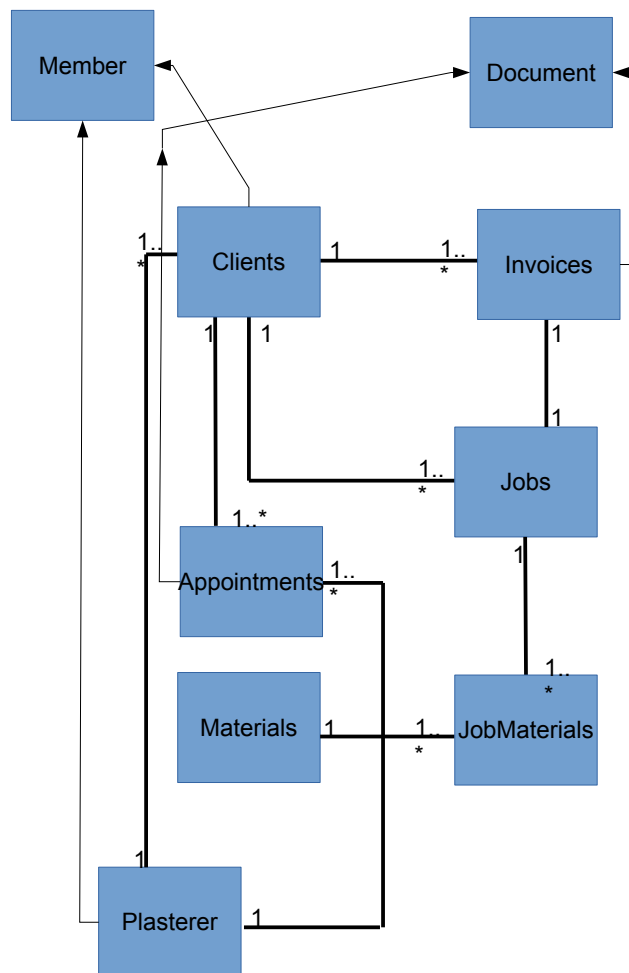
Figure 2.18: Object Relationships

2.4.4   Class Definitions

Key:

Label
Attributes
Behavious

Member
MemberID
MemberTitle
MembeFirstName
MemberSurname
MemberAddrLine1
MemberAddrLine2
MemberAddrLine3
MemberAddrLine4
MemberEmail
MemberPhoneNumber
AddMemberTitle
AddMemberFirstName
AddMemberSurname
AddMemberAddrLine1
AddMemberAddrLine2
AddMemberAddrLine3
AddMemberAddrLine4
AddMemberEmail
AddMemberPhoneNumber
EditMemberTitle
EditMemberFirstName
EditMemberSurname
EditMemberAddrLine1
EditMemberAddrLine2
EditMemberAddrLine3
EditMemberAddrLine4
EditMemberEmail
EditMemberPhoneNumber

Client: extends **Member**

Plasterer: extends **Member**
PlastererDailyRate
AddPlastererDailyRate
EditPlastererDailyRate

Job:

JobID
ClientID
PlastererID
JobDescription
JobAddrLine1
JobAddrLine2
JobAddrLine3
JobAddrLine4
JobDaysWorked
JobComplete
JobPaid
InvoiceID

AddJobDescription
AddJobAddrLine1
AddJobAddrLine2
AddJobAddrLine3
AddJobAddrLine4
AddJobDaysWorked
EditJobComplete
EditJobPaid
EditJobDescription
EditJobAddrLine1
EditJobAddrLine2
EditJobAddrLine3
EditJobAddrLine4
EditJobDaysWorked

Material:

MaterialID
MaterialName
MaterialPrice

AddMaterialName
AddMaterialPrice

EditMaterialName
EditMaterialPrice

Document:
DocumentID
DocumentDate
DocumentTime
DocumentText
AddDocumentDate
AddDocumentTime
AddDocumentText
EditDocumentDate
EditDocumentTime
EditDocumentText

Invoice: extends **Document**
ClientID
JobID
PlastererID
InvoiceAmountPreTax
InvoiceAmountAfterTax
InvoiceReceived
AddInvoiceAmountPreTax
AddInvoiceAmountAfterTax
AddInvoiceReceived
EditInvoiceAmountPreTax
EditInvoiceAmountAfterTax
EditInvoiceReceived

Appointment: extends **Document**
ClientID
PlastererID
AppointmentAddrLine1
AppointmentAddrLine2
AppointmentAddrLine3
AppointmentAddrLine4
AddAppointmentAddrLine1
AddAppointmentAddrLine2
AddAppointmentAddrLine3
AddAppointmentAddrLine4
EditAppointmentAddrLine1
EditAppointmentAddrLine2
EditAppointmentAddrLine3
EditAppointmentAddrLine4

2.5   Prototyping

PyQt4 Printing Prototyping

I researched and developed a small printing prototype using PyQt4 and
Python3. I found information on printing with PyQt4 in the book *Rapid GUI
Programming with Python and Qt* By *Mark Summerfield*. The book described
how you can print using a QTextDocument, QPrinter and HTML formatting.
Below is the code for this prototype.

```python
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys


class MainWindow(QMainWindow):
    """ This is a test for the printing function """

    def __init__(self):
        super().__init__()


        self.setWindowTitle("Printing Test")
        self.printer = QPrinter()
        self.printer.setPageSize(QPrinter.Letter)
        self.mainLayout()


    def mainLayout(self):

        self.layout = QHBoxLayout()
        self.printButton = QPushButton("Print")

        self.layout.addWidget(self.printButton)

        self.mainWidget = QWidget()
        self.mainWidget.setLayout(self.layout)

        self.setCentralWidget(self.mainWidget)

        self.printButton.clicked.connect(self.printViaHtml)

    def getCurrentDate(self,dateFormat):
        date =
            QDate.currentDate().toString(dateFormat)
        return date
```

```python
36
37      def statementHtml(self):
38
39          companyName = "DnA Plastering"
40          date = self.getCurrentDate("dd.MM.yyyy")
41
42          jobItems = [["25KG
                Plaster","30.00","5","150.00"],["Angle
                Beading 3M","7.00","5","35.00"]]
43
44          invoiceId = "4564"
45          amountDue = 0.0
46          for each in jobItems:
47              price = float(each[3])
48              amountDue += price
49          amountAfterTax = amountDue * 1.2
50          address = ["15 The
                Glebe","Haverhill","Suffolk","CB9 0DL"]
51
52
53
54
55          html = u""
56          html += ("<h1 align='center'>{0}</h1>"
57                  "<table width='30%' align='left'
                        cellpadding='10px'>").format(companyName)
58
59          for each in address:
60              html +=
                    ("<tr><td>{0}</td></tr>").format(each)
61
62
63          html += ("</table>"
64                  "<table width='30%' border='1'
                        align='right' cellpadding='10px'>"
65                  "<tr><td><b>Invoice</b>
                        #</td><td>{0}</td></tr>"
66                  "<tr><td><b>Date</b></td><td>{1}</td></tr>"
67                  "<tr><td><b>Amount
                        Due</b></td><td>{2}</td></tr>"
68                  "</table>"
69                  "<br><hr/>"
70                  "<table width='100%' border='1'
                        cellpadding='10px'>"
71                  "<tr><td><b>Material/Item</b></td><td><b>Unit
                        Cost</b></td><td><b>Quantity</b></td><td><b>Price</b></t
```

```python
72
73          for item in jobItems:
74              html +=
                    ("<tr><td>{0}</td><td>{1}</td><td>{2}</td><td>{3}</td></tr>")
75
76          html += ("<tr><td></td><td></td><td><b>Sub
                Total</b></td><td>{0}</td></tr>"
77                  "<tr><td></td><td></td><td><b>Total
                        (20% VAT
                        added)</b></td><td>{1}</td></tr>").format(amountDue,amou
78
79          html += ("</table>"
80                  "<br>"
81                  "<hr/>"
82                  "<p
                        align='center'>{0}</p>").format(companyName)
83
84
85          return html
86
87
88
89      def printViaHtml(self):
90
91          html = self.statementHtml()
92
93          dialog = QPrintDialog(self.printer, self)
94          if dialog.exec_():
95              document = QTextDocument()
96              document.setHtml(html)
97              document.print_(self.printer)
98          else:
99              print("The print process has failed!")
100
101         print(html)
102
103
104
105 if __name__ == "__main__":
106
107     app = QApplication(sys.argv)
108     window = MainWindow()
109     window.show()
110     window.raise_()
111     app.exec_()
```

2.6   Definition of Data Requirements

2.6.1   Identification of all data input items

2.6.2   Identification of all data output items

2.6.3   Explanation of how data output items are generated

2.6.4   Data Dictionary

2.6.5   Identification of appropriate storage media

## 2.7   Database Design

2.7.1   Normalisation

ER Diagrams
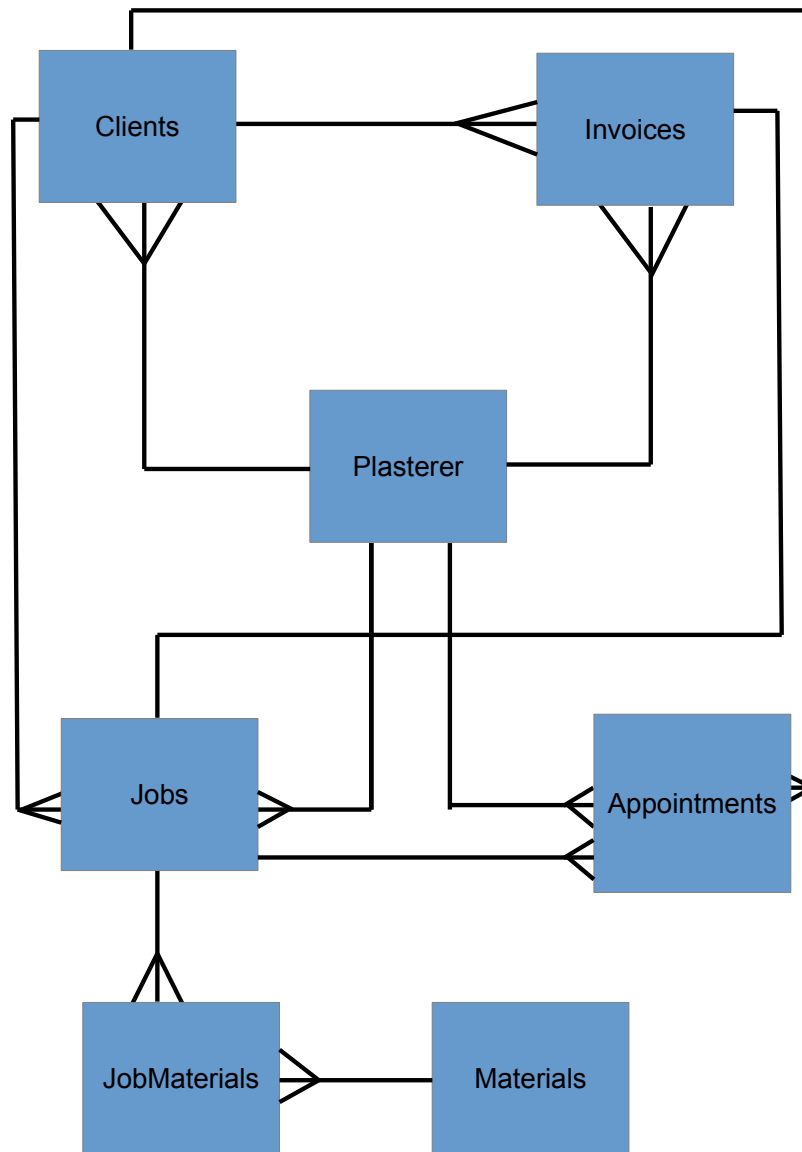
Figure 2.19: This is the entity relationship diagram for the sqlite3 database.

Entity Descriptions

Below are the entity descriptions for the various entites in the proposed system. An <u>underlined</u> attribute denotes a primary key and a *emphasised* attribute signifies a foreign key in the entity.

Client(<u>ClientID</u>, ClientTitle, ClientFirstName, ClientSurname, ClientAddrLine1, ClientAddrLine2, ClientAddrLine3, ClientAddrLine4, ClientEmail, ClientPhoneNumber)

Plasterer(<u>PlastererID</u>, PlastererFirstName, PlastererSurname, PlastererAddrLine1, PlastererAddrLine2, PlastererAddrLine3, PlastererAddrLine4, PlastererEmail,PlastererPhoneNumber, PlastererDailyRate)

Job(<u>JobID</u>, *ClientID*, *PlastererID*, JobDescription, JobAddrLine1, JobAddrLine2, JobAddrLine3, JobAddrLine4, JobDaysWorked, JobComplete, *InvoiceID*)

Material(<u>MaterialID</u>,MaterialName,MaterialPrice)

JobMaterials(<u>JobMaterialsID</u>, *JobID*, *MaterialsID*, JobMaterialsQuantity)

Invoice(<u>InvoiceID</u>, *JobID*, InvoiceAmountPreTax, InvoiceAmountAfterTax, InvoiceReceived, InvoiceDate, InvoiceText, InvoicePaid)

Appointment(<u>AppointmentID</u>, *JobID* AppointmentDate, AppointmentTime)

1NF to 3NF

1NF
Non-Repeating Attributes
PersonID (Primary Key)
ClientTitle
ClientFirstName
ClientSurname
ClientAddrLine1
ClientAddrLine2
ClientAddrLine3
ClientAddrLine4
ClientEmail
ClientPhoneNumber
PlastererTitle
PlastererFirstName
PlastererSurname
PlastererAddrLine1
PlastererAddrLine2
PlastererAddrLine3

PlastererAddrLine4
PlastererEmail
PlastererPhoneNumber
PlastererDailyRate
Repeating Attributes
JobID (Primary Key)
PersonID (Composite Key)
JobDescription
JobAddrLine1
JobAddrLine2
JobAddrLine3
JobAddrLine4
JobDaysWorked
JobComplete
MaterialName
MaterialPrice
JobMaterialsQuantity
InvoiceAmountAfterTax
InvoiceAmountPreTax
InvoiceReceived
InvoiceDate
InvoiceText
InvoicePaid
AppointmentDate
AppointmentTime
AppointmentAddrLine1
AppointmentAddrLine2
AppointmentAddrLine3
AppointmentAddrLine4

2NF
Group
PersonID (Primary Key)
ClientTitle
ClientFirstName
ClientSurname
ClientAddrLine1
ClientAddrLine2
ClientAddrLine3
ClientAddrLine4
ClientEmail
ClientPhoneNumber
PlastererTitle

| PlastererFirstName |
| PlastererSurname |
| PlastererAddrLine1 |
| PlastererAddrLine2 |
| PlastererAddrLine3 |
| PlastererAddrLine4 |
| PlastererEmail |
| PlastererPhoneNumber |
| PlastererDailyRate |
| **Group** |
| JobID (Primary Key) |
| PersonID (Composite Key) |
| JobDescription |
| JobAddrLine1 |
| JobAddrLine2 |
| JobAddrLine3 |
| JobAddrLine4 |
| JobDaysWorked |
| JobComplete |
| **Group** |
| JobID (Primary Key) |
| MaterialName |
| MaterialPrice |
| JobMaterialsQuantity |
| **Group** |
| PersonID (Primary Key) |
| InvoiceAmountAfterTax |
| InvoiceAmountPreTax |
| InvoiceReceived |
| InvoiceDate |
| InvoiceText |
| InvoicePaid |
| AppointmentDate |
| AppointmentTime |
| AppointmentAddrLine1 |
| AppointmentAddrLine2 |
| AppointmentAddrLine3 |
| AppointmentAddrLine4 |

| **3NF** |
| **Group** |
| PersonID (Primary Key) |
| ClientTitle |

ClientFirstName
ClientSurname
ClientAddrLine1
ClientAddrLine2
ClientAddrLine3
ClientAddrLine4
ClientEmail
ClientPhoneNumber
PlastererTitle
PlastererFirstName
PlastererSurname
PlastererAddrLine1
PlastererAddrLine2
PlastererAddrLine3
PlastererAddrLine4
PlastererEmail
PlastererPhoneNumber
PlastererDailyRate
Group
JobID (Primary Key)
PersonID (Composite Key)
JobDescription
JobAddrLine1
JobAddrLine2
JobAddrLine3
JobAddrLine4
JobDaysWorked
JobComplete
Group
JobID (Primary Key)
MaterialID (Foreign Key)
JobMaterialsQuantity
Group
MaterialID (Primary Key)
MaterialName
MaterialPrice
Group
InvoiceID (Primary Key)
InvoiceAmountAfterTax
InvoiceAmountPreTax
InvoiceReceived
InvoiceDate
InvoiceText
InvoicePaid
Group

AppointmentID (Primary Key)
AppointmentDate
AppointmentTime
AppointmentAddrLine1
AppointmentAddrLine2
AppointmentAddrLine3
AppointmentAddrLine4
Group
PersonID (Primary Key)
AppointmentID (Foreign Key)
InvoiceID (Foreign Key)

2.8   Security and Integrity of the System and Data

2.8.1   Security and Integrity of Data

2.8.2   System Security

2.9   Validation

2.10   Testing

2.10.1 Outline Plan

Test Series	Purpose of Test Series	Testing Strategy	Strategy Rationale
Example	Example	Example	Example

2.10.2 Detailed Plan

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
Example	Example	Example	Example	Example	Example	Example	Example

Chapter 3

Testing

3.1 Test Plan

3.1.1   Original Outline Plan

Test Series	Purpose of Test Series	Testing Strategy	Strategy Rationale
Example	Example	Example	Example

3.1.2   Changes to Outline Plan

Test Series	Purpose of Test Series	Testing Strategy	Strategy Rationale
Example	Example	Example	Example

3.1.3   Original Detailed Plan

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
Example	Example	Example	Example	Example	Example	Example	Example

3.1.4   Changes to Detailed Plan

Test Series	Purpose of Test	Test Description	Test Data	Test Data Type (Normal/ Erroneous/ Boundary)	Expected Result	Actual Result	Evidence
Example	Example	Example	Example	Example	Example	Example	Example

3.2 Test Data

3.2.1 Original Test Data

3.2.2 Changes to Test Data

3.3 Annotated Samples

3.3.1 Actual Results

3.3.2 Evidence

3.4   Evaluation

3.4.1   Approach to Testing

3.4.2   Problems Encountered

3.4.3   Strengths of Testing

3.4.4   Weaknesses of Testing

3.4.5   Reliability of Application

3.4.6   Robustness of Application

Chapter 4

System Maintenance

4.1 Environment

4.1.1 Software

4.1.2 Usage Explanation

4.1.3 Features Used

4.2 System Overview

4.2.1 System Component

4.3 Code Structure

4.3.1 Particular Code Section

4.4 Variable Listing

4.5 System Evidence

4.5.1 User Interface

4.5.2 ER Diagram

4.5.3 Database Table Views

4.5.4 Database SQL

91

4.5.5 SQL Queries

4.6 Testing

4.10.1 Module 1

Chapter 5

User Manual

5.1   Introduction

5.2   Installation

5.2.1   Prerequisite Installation

Installing Python

Installing PyQt

Etc.

5.2.2   System Installation

5.2.3   Running the System

5.3   Tutorial

5.3.1   Introduction

5.3.2   Assumptions

5.3.3   Tutorial Questions

Question 1

Question 2

94

5.3.4   Saving

5.3.5   Limitations

5.4   Error Recovery

Chapter 6

Evaluation

6.1 Customer Requirements

6.1.1 Objective Evaluation

6.2 Effectiveness

6.2.1 Objective Evaluation

6.3 Learnability

6.4 Usability

6.5 Maintainability

6.6 Suggestions for Improvement

6.7 End User Evidence

6.7.1 Questionnaires

6.7.2 Graphs

6.7.3 Written Statements