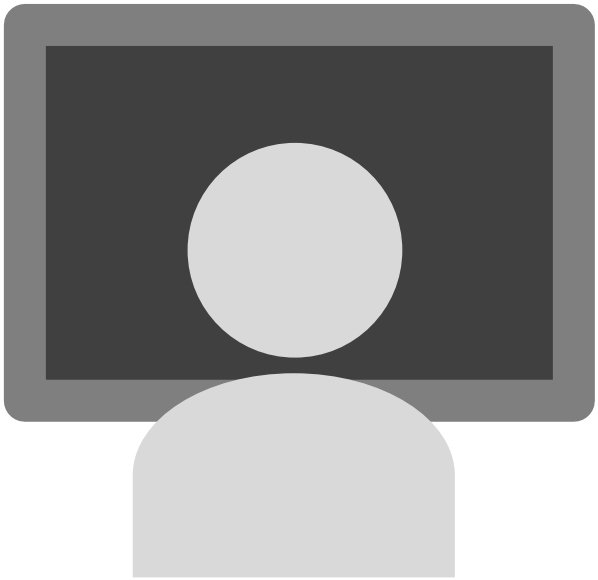
Zig Zag Education

Computer Science A Level | AQA | 7517

2015 specification
first assessment in 2017



Non-Exam Assessment Companion

for A Level AQA Computer Science

J Dix

zigzageducation.co.uk POD 7758

Contents

Thank you for choosing ZigZag Education!

Talk to Us!

Love it as it is?
Let the author and other teachers know what you think

Got a suggestion?
If your improvement leads to an update we will send you a new copy for free

Found a problem?
We will fix it and send you a free updated copy

We ♥ your feedback!
Let us know what you think using the feedback sheet on the next page.
£10 ZigZag Voucher for detailed & complete reviews!

Web: **zzed.uk/more**

Email: **computerscience @zigzageducation.co.uk**

Real Person: **0117 950 3199**

Fax: **0117 959 1695**

Post: **ZigZag Education, Unit 3, Greenway Business Centre, Doncaster Road, Bristol BS10 5PY**

Become a Published Author

ZigZag is a large community of over 6000 teachers & educationalists.
Review new titles, publish your own work or write to a brief.

Fancy being involved?

Then register at...

publishmenow.co.uk

The Professional Publishing Community

Alternatively email new resource ideas directly to...
publishmenow@zigzageducation.co.uk

For more resources go to **zzed.uk/more**
where you can preview every page before you buy

Teacher Feedback Opportunity

£10 ZigZag Voucher for detailed & complete reviews! ◆ Use for problems/areas for improvement/positive feedback

Resource ID & Name	7758 Non-Exam Assessment Companion for A Level AQA Computer Science

School Name	

Your Name		**Position**	

Overall, what did you think about this resource?.....

.....

.....

.....

I particularly like this resource because… .....

.....

.....

How does it help you or your students? .....

.....

It is better than some other resources because… .....

.....

.....

What might you say to a colleague in a neighbouring school to persuade them to use this resource?.....

.....

.....

.....

How well does it match your specification (and which specification is this)?.....

.....

Other comments, suggestions for improvements, errors found (please give page numbers) etc. .....

.....

.....

Resources I would like published:	

Resources I might write*, or have written,* for consideration for publication:	

Fax to:	Email to:	Submit online:	Post to:
0117 959 1695	**feedback@ zigzageducation.co.uk**	**zzed.uk/feedback**	**ZigZag Education, Unit 3, Greenway Business Centre, Doncaster Road, Bristol BS10 5PY**

INTERNAL USE ONLY
Feedback logged: ✓☐
Complete & detailed: Y /N
If detailed, £10 sent: ✓☐

Terms and Conditions of Use

Terms and Conditions

Please note that the **Terms and Conditions** of this resource include point 5.3, which states:

> **"You acknowledge that you rely on your own skill and judgement in determining the suitability of the Goods for any particular purpose."**

"We do not warrant: that any of the Goods are suitable for any particular purpose (e.g. any particular qualification), or the results that may be obtained from the use of any publication, or expected exam grades, or that we are affiliated with any educational institution, or that any publication is authorised by, associated with, sponsored by or endorsed by any educational institution."

Copyright Information

Every effort is made to ensure that the information provided in this publication is accurate and up to date but no legal responsibility is accepted for any errors, omissions or misleading statements. It is ZigZag Education's policy to obtain permission for any copyright material in their publications. The publishers will be glad to make suitable arrangements with any copyright holders whom it has not been possible to contact.

Students and teachers may not use any material or content contained herein and incorporate it into a body of work without referencing/acknowledging the source of the material ("Plagiarism").

Disclaimers

ZigZag Education is not affiliated with Pearson, Edexcel, OCR, AQA, WJEC, Eduqas, EA, International Baccalaureate Organization or DFE in any way nor is this publication authorised by, associated with, sponsored by or endorsed by these institutions unless explicitly stated on the front cover of this publication.

Different teachers, Heads of Departments and Moderators have different personal views on what information and support to provide an individual or group for a given specification and when to provide this. Different specifications and modules require different levels of support or differing amounts of information to be provided, or they prohibit information or support to be given to a student above a certain level. For very high level work no support or information may be appropriate or a required feature of the module.

Where the teacher uses any of the material from this resource to support coursework, controlled assessment or similar then the teacher must ensure that they are happy with the level of information and support provided pertaining to their personal point of view and to the constraints of the specification and to others involved in moderation or the process or delivery of the course. It is considered essential that the teacher adapt, extend and or censor any parts of the contained material to suit their needs, the needs of the specification, the needs of moderators and the needs of the individual or group concerned. As such, the teacher must determine which parts of the material, if any, to provide to students and which parts to use as background information for themselves.

In this matter they should also determine the nature of information or support provided, taking into consideration the medium and style of support. So specifically, they should determine which information to provide verbally, if any, and which, if any, to provide in written format. Similarly, if the style of the support is inappropriate but the information or support itself is appropriate, then the material should be so adjusted to achieve this end. For example, if any of the contained material appears to overly direct the student, which may result in a downgrading of the student's project, then the teacher should remove or adapt this material to avoid this unless it is the teacher's intention to do this to achieve a certain level within the project.

In summary, it is intended that these materials be used appropriately and at the teacher's own discretion, and that the teacher take into consideration the ability of the individual or group. It is the teacher's responsibility to assess the suitability of coursework/project publications and to decide which pages, if any, to hand out to students.

Links to other websites, and contextual links are provided where appropriate in ZigZag Education publications. ZigZag Education is not responsible for information on sites that it does not manage, nor can we guarantee, represent or warrant that the content contained in the sites is accurate, legal and inoffensive, nor should a website address or the inclusion of a hyperlink be taken to mean endorsement by ZigZag Education of the site to which it points. This includes websites that users are directed to via the convenient ZZed.uk short URLs.

Teacher's Introduction

IMPORTANT – please read before using this resource

This resource is intended to supplement your teaching only. As with all Non-Exam Assessment (NEA) materials it is the teacher's responsibility to decide what level of support is appropriate for their students and in accordance with the rules from the exam board. For example, you may simply wish to read this material to better inform yourself. Alternatively, you may consider whether it is appropriate to distribute some of the material to students for reference.

The resources here are provided as one experienced teacher's interpretation of the specification. The author does not have any special knowledge of what to expect on any particular assessment.

All exemplar material in this resource is based on entirely original, fictitious scenarios. Any possible resemblances to any future task released by AQA or any other exam board is co-incidental. However, we remind you that it is the teachers' responsibility to decide how this resource can be used to support your students.

The Non-Examined Assessment (NEA) component of the AQA A Level in Computer Science is designed as an opportunity for candidates to display their programming skill by creating a piece of software on a topic that interest them. This may be a problem to solve or a programmed investigation of a computer science concept.

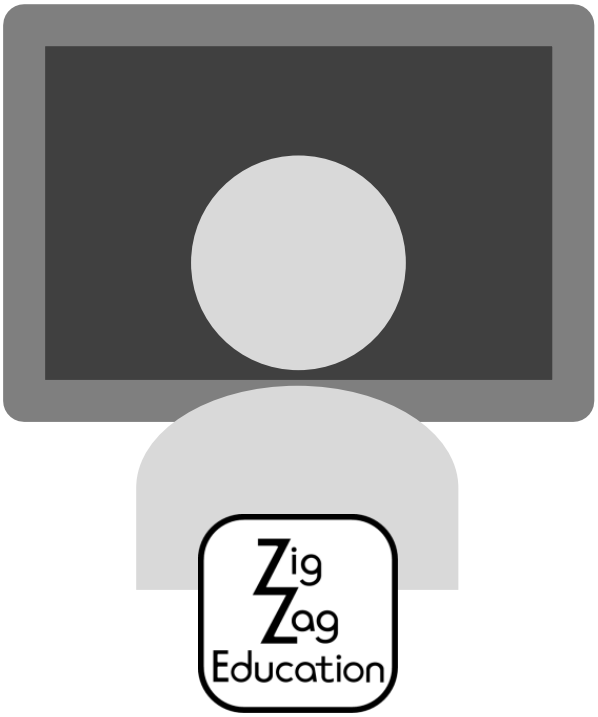
This component is worth 20% of the overall A Level and is a major piece of work.

This guide is intended to support students produce the required documentation for their NEA, which is marked by the teacher and moderated by the exam board. It attempts to achieve this by providing a breakdown of the marking criteria, with tips, examples and checklist prompts given throughout.

It does not replace the role of the teacher or tutor who may need to supplement the advice contained herein in the context of their own students' work.

When photocopying the resource, it is useful if the notes page (found at the back of the guide) is photocopied multiple times and placed at the end of each section of the guide. This is then a useful place for students, and possibly teachers / tutors, to add their own notes or comments.

J Dix



A Level AQA Computer Science

Non-Exam Assessment Guide

Section	Deadline
Project Idea	.....
Formal investigation	.....
Analysis	.....
Design	.....
Technical Solution	.....
Testing	.....
Evaluation	.....

1. Choosing Your Project

In many respects this is the most important part of your project and you should take note of the advice below. First and foremost though make sure that you choose a project that interests you and preferably relates to something that you already have some knowledge of.

1.1 – Types of project

A data-processing project	This is a type of project that will insert, update, delete, retrieve and process data from some kind of data storage file (typically a database, DAT files or text files). Typically it will link to a data storage file through a GUI interface (through the use of a programming language) or web interface (through the use of server-side scripting). Typical projects of this nature include: • **Booking Systems** (Appointments, Seats, products and services etc.) • **Stock Control** systems • **Service systems** (to enter, allocate and monitor service requests –,e.g. printing, reprographics, technical assistance) • **Information systems** (this can also include the use of APIs, live feeds to retrieve and feed in live data. It is also possible to use live twitter feeds (for example) to accumulate a large data set for further investigation or to model a scenario) • **Revision/ testing programs for students.** This can be done for any subject and allows users to take quizzes or tests, reinforces key teaching points and stores scores and other performance data.
Mathematical Projects	If you choose to do a mathematical project then the mathematical processing you are modelling should be to at least KS5 standard. Projects of this nature might include graphing, trigonometry, decision maths, etc.
Simulations	This would be a program that simulates a real world event. Examples of this type of project are orbital simulations, environmental simulations, financial simulations and time based organisational simulations (e.g. queuing simulations).
Computer Games	Bound to be a very popular option but you must ensure that your game is all about the programming of it and you don't spend all your time producing beautiful graphics that don't do anything. Also, seek clear guidance from your project supervisor when setting out objectives for your game (what it will do). It is very common for students not to finish computer game projects because the scope of the objectives is too large to fit into the available time-frame.
Control Systems	A system that controls a robot, or another device. **https://www.arduino.cc/en/Guide/Introduction**
Websites with dynamically changing content	These will normally connect to remote databases (online shops, online ordering, online booking, online information systems etc.) It must be for a client-server model and should have significant processing. There are no marks awarded for your skill at writing HTML as this is not a programming language.
Optimisation problems	These types of projects might use algorithms to compile the most efficient routes around an area, the most efficient rota for an organisation, timetabling issues etc.
Apps for mobile devices	You need to ensure there are considerable amounts of potential processing in the app you want to build. Additionally you should avoid development environments that use drag and drop components and do a significant amount of the programming for you.
Artificial intelligence	These can be projects like games where a player plays against the computer. Additionally it is possible to develop an expert system that allows non- expert users to navigate a knowledge base. Such systems are often used in call-handling centres to both route and handle calls.

Investigatory Projects	This is where you choose an area of a subject you are interested in and you conduct a programmed investigation of this area. Examples of this type of project might be: • Big Data – analysing live data feeds such as Twitter • Exploring large publicly available data sets (such as the World Bank) • Machine Learning algorithms • Maze problems • Traversal algorithms • Encryption algorithms

1.2 – Choosing a project within your skill set

You must consider whether or not the problem you choose will give you the opportunity to demonstrate the level of technical skill you are capable of. The solution you come up with must be **a fully-coded solution** (all processing is done with code that you have written yourself). The biggest section of marks is for the technical solution and in order to get top marks in this section you will need to demonstrate a high degree of programming competence. Table 1 below shows the kind of skills that get you those marks. It is not intended to be an exhaustive list nor is it simply a tick list of things you must do – it is intended as a guide to judging skill level. Your teacher will be able to give you firm guidance about which mark band a particular feature of your code might fall into.

In reality you want to make sure that you will be able to use a range of skills that **at least** falls into the middle mark band as just using skills from the lower mark band will severely restrict the amount of marks you can be awarded for your technical solution. Very competent programmers should of course aim at the top mark band technique.

When deciding which mark band your technical solution falls into, your teacher will apply a "best fit" policy.

Table 1 – Skills List

Top Mark band (very good programmer)	**Middle mark band** (fair programmer)	**Lower Mark band** (some programming skill)
Include a range of these skills • Candidate written classes (OOP) (including inheritance, aggregation etc.) • OOP objects created at runtime • Complex standard algorithms and structures (structures, arrays of records, stacks, queues, linked lists, trees and graphs) • Use of regular expressions • Use of hashing techniques • Use of recursion • Merge sorting or equivalent • Implementing complex mathematical processes (KS5 Maths at least) • Time simulation / scheduling • Complex client server model **Specific to Data- handling solutions** • A complex data model with many related tables (multiple normalised entities to at least 3rd normal form) • Using parameterised cross-table SQL statements embedded in the programming language • User generated Data definition statements (CREATE TABLE embedded in the code	**Include a range of these skills** • Use of arrays (2-D) • Some standard algorithms like bubble sorts and binary searches • Simple client-server model • Some simple OOP classes **Data- handling solutions** • Simple data model with a few inter-linked tables • Single-table SQL statements • Use of text files for storing and retrieving data	• 1-D arrays • Uses a small range of different data types • Uses linear searching methods • Uses simple mathematical calculations • Accesses table data without the use of SQL

1.3 – Getting a real end-user

There is no formal requirement that you have an end-user, real or otherwise. However, it is a good idea to find someone who can act as your end-user, i.e. the person who commissions your project because they have a problem(s) for you to solve with a programmed solution. They can give you valuable **realistic feedback** on your design and solution. You do need convenient access to them so you can ask questions about the problem and get feedback. However this can be done through a variety of electronic methods (email, Skype, text, online forms) as well as face to face. Additionally it is a good idea to have contact with some potential users of your system, allowing them to give you feedback which can be used to help frame objectives and design.

1.4 – Choosing the right programming language

This is quite simple really; choose a language that you are familiar with. It is true that some students decide to learn a new language in order to use it for their project but if you do this then you will need to factor in extra time.

1.5 – Getting approval from your teacher

Your teacher should give you advice about a number of issues so do make use of their expertise when choosing a project. They can tell you:

- whether your project ideas are within your programming skill set
- whether your project can be achieved within the timescale and if not give you advice about reducing the scope of it.
- whether your project is substantially similar to another student's project in your cohort. It is a requirement that the problem you choose should be different from that chosen by others so that there is no possibility of collusion.

1.6 – How you will be assessed

Your project is assessed through the use of a written report. The report is split into a number of sections.

Section Title	Max Mark
Analysis	9
Documented Design	12
Technical Solution	42
Testing	8
Evaluation	4
Total	**75**

Your teacher will set regular deadlines for the various sections of work to be completed by. Use the booklet cover to fill in those deadlines so that you don't miss them.

1.7 – First steps

The first step is to do decide what type of project you want to do and find an end-user. Of course sometimes it happens the other way round, you get an end-user and that determines the type of project you do. Whichever way it is for you, your teacher will normally ask you to complete some kind of project proposal (or you can use the one below). It is a good idea to try and find a couple of alternatives initially, just in case one of your ideas is substantially similar to another student's proposal.

Project Proposal Form

Name:	**Class:**
Type of Project:	
Project title (Working title):	
Potential end-user(s) and the organisation they represent:	
Outline description of your project idea:	
What problems does the existing system have that might need a potential computer solution?	
What kind of access do you have to your potential end-user?	
What potential programming skills (from Table 1) will it enable you to demonstrate	

Alternative Project idea

Name:	**Class:**
Type of Project:	
Project title (Working title):	
Potential end-user(s) and the organisation they represent:	
Outline description of your project idea:	
What problems does the existing system have that might need a potential computer solution?	
What kind of access do you have to your potential end-user?	
What potential programming skills (from Table 1) will it enable you to demonstrate	

2. Analysis

You should think of Analysis as an opportunity! It is your opportunity to explain the scope of your project to the marker / moderator and this needs to be done in a variety of ways of which the important areas are:

1. *An **investigation of the project or problem area** (interviews, observations, questionnaires, research and examining documents)*
2. *A **description of the problem(s) of the existing system** for which your end-user is wanting a solution (or a new solution)*
3. *A **modelling of the project area or how the existing system actually works** (before you start designing and building a new system to solve its problems). This will involve the use of diagrams that model how aspects of the system work.*
4. *A list of **numbered and SMART objectives** (**what the new system must do**) that arise out of the investigation you have done. This should include both qualitative and quantitative objectives*
5. *A list of **performance criteria** for your objectives that include specific details of how you will judge the success of each objective*
6. *Some **consideration of the different types of solution** that might be available to fulfil your objectives.*

2.1 – Investigating the project problem

The investigation is the section of your analysis where you find the information you need to complete the rest of the analysis so it needs to be done thoroughly. The assessment criteria clearly state that there must be evidence of dialogue with an end-user or potential users that helps to frame the requirements of the system you will build.

Preparation for the Investigation

Before you embark on this section you should ensure you have two very important items firmed up:

1. You have found **a problem to solve** and someone who can act as your end-user. Delays in contacting your end user at this stage can have a serious impact on timescales if you are relying on their feedback to make progress. You may also need access to other users of the system so make sure that this won't be a problem either.
2. Your NEA supervisor (your tutor or teacher) has **approved your project**. Your teacher or tutor will have a good idea of whether your project idea will give you access to a mark range that is reflective of your ability and can be realistically achieved within the allowed timescale – so you should always run your idea past them first before you embark on your formal investigation.

What methods should I choose?

There are a number of methods that can be used to investigate the project problem and the existing system (if there is one). You must select the methods that are appropriate to your end-user, other users and the project. In general a good investigation will have used a variety of methods to find the required information that will help you complete the other sections of your analysis.

- **Interviews**
  This is often the most common way of investigating that students tend to use and may be used with your actual end-user as well as with other potential users of the system. You prepare a list of questions, arrange a date and then ask your prepared questions noting down the answers as your end user gives you them. Although interviews are normally face to face it is acceptable to conduct interviews over the phone or through applications like Skype.

  If other issues (that you have not anticipated with a question) come out during the interview then these can always be added at the end of your interview responses.

An extract from a competed interview is shown in Fig 1

Figure 1 – Classic Furnishings: Interview

Interview with Ms End User
Sole Proprietor of Classic Furnishings

Date of interview – 22/10/17
Place of interview – End user's home (address)

Q1 – Can you give me some background information about your business?
Classic furnishings is run just by me. I have run this business from my own home for more than 10 years. I do not employ any other people. I design, make and fit mostly curtains and blinds, although I occasionally make other soft furnishing items like bed covers and cushion covers on request. All soft furnishing items are bespoke and made to the individual customer's requirements.

Q2 – What happens when you receive an order?
Customers ring up and ask me to come out and see them so that I can measure up, discuss their requirements and show them my sample books of suitable materials.

I take with me my order book, my books of sample material and my catalogue of trimmings and fittings. I write down the customer's requirements in my order book. Every customer's order is put onto a new page in my order book. When I get home I do some calculations and write down the projected cost onto the appropriate order page. Form this I then word process a quotation to send to the customer.

- **Observations**
  This is a useful method if your end user or other users find it difficult to describe the methods they use. Again in preparation you should write up a series of observation points (to focus your mind on what you are looking for from the observation), arrange a date for the observation and then as you are conducting the observation write down notes on what you observe under the appropriate observation points.

  Extracts from an observation are shown in Fig 1 and Fig 2. Fig 2 is the empty template ready for notes to be written down during the observation and Fig 3 is an extract from the competed observation notes.

Figure 2 – Classic Furnishings: Observation Template Example

Observation of Ms End User (Sole Proprietor of Classic Furnishings) visiting a Customer

Date of observation – 22/10/17
Place of observation – Mrs Smith home (address)

Point 1 – What does the end user do (order of events)?

Point 2 – What does the end user record and where?

Figure 3 – Classic Furnishings: Example Interview

Observation of Ms End User (Sole Proprietor of Classic Furnishings) visiting a customer

Date of observation – 22/10/17
Place of observation – Mrs Smith home (address)

Point 1 – What does the end user do (order of events)?

- Greets the customer and spends some time chatting about what ideas Mrs Smith has and what items she wants making
- Measures up the lounge window (height and width)
- Draws a sketch of the window in the order book – this seems to help her note any additional or unusual features of the window. Adds the measurements to the sketch.
- Asks Mrs Smith what fullness of curtain material she desires. This is about how much gathering she wants in the curtains.
- Mrs Smith then selects an appropriate curtain material and trimmings
- Then end user makes a note of Mrs Smith's selection in her order book

Point 2 – What does the end user record and where?

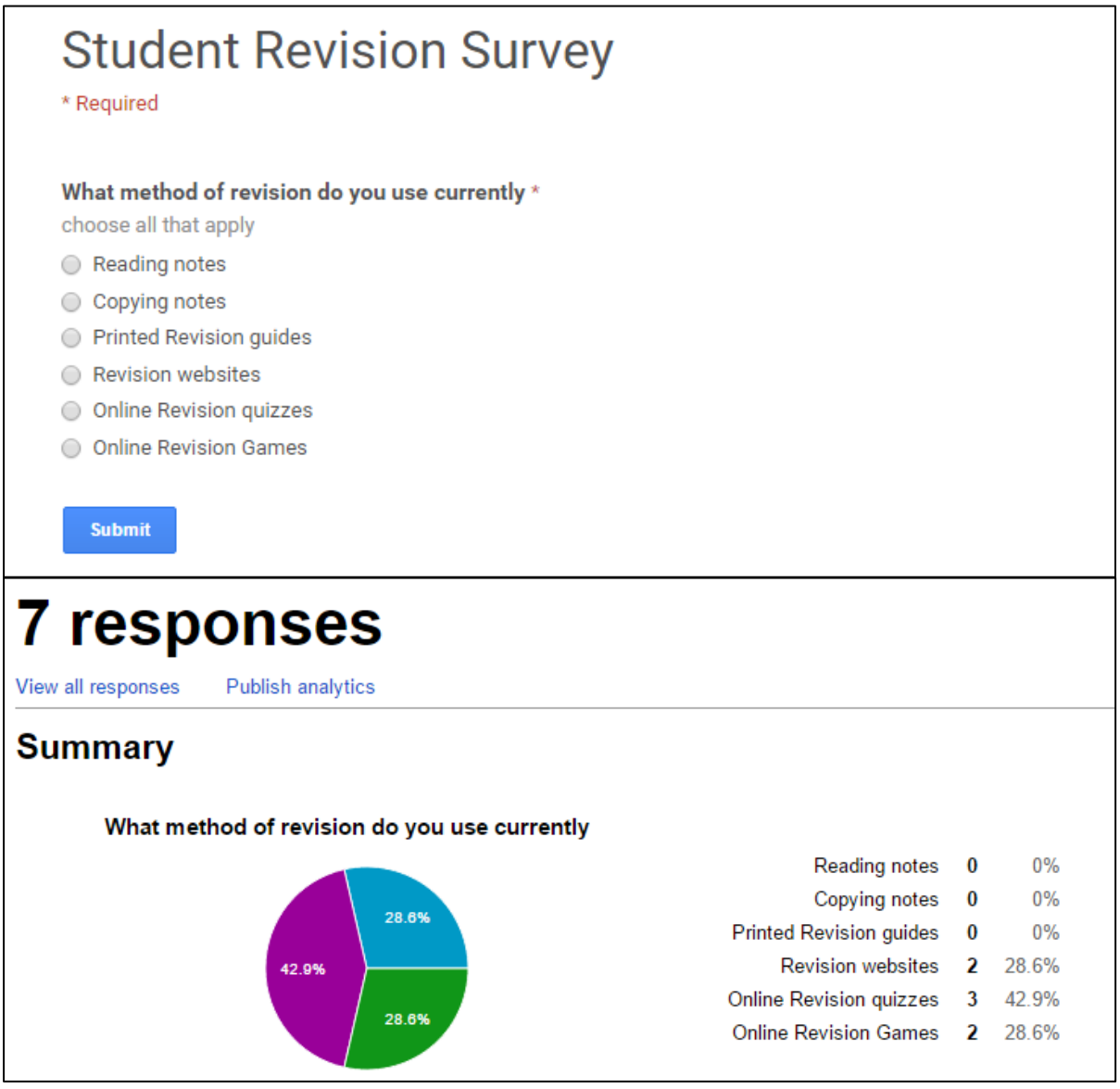
- Customer first name and surname
- Customer contact number
- Sketch of the window
- Dimensions:
  a) Height (drop)
  b) Width (of the window)
  c) Fullness (gathering)
  d) Depth
- Material number
- Material name
- Material cost per metre

All noted in the order book

- **Questionnaires / Online Surveys**
  These are particularly useful where you may have a largish number of potential users to obtain information from. It is also possible to quantify the responses from certain questions and represent the findings in the form of charts and graphs. There are many online tools that you can use to both build your questionnaire and quantify responses for you.

  Fig 4 and 5 show extracts from an online survey that students were asked to complete to find out which method of revision they favoured the most

Figures 4 & 5 – Classic Furnishings: Online Survey



- **Examining Documents**
  This will involve looking at any documentation that the existing system uses. It is particularly useful for finding data items that are input into the system (where data collection forms are used)

- **Research**
  This might be useful as additional evidence of the nature of a problem. For example – A popular data-handling project is providing online or electronic revision systems for students at various levels. Your end user might justify their need for a new revision system by explaining the shortcomings of what is already available online. You could therefore do some research on online revision offerings to help you point out and explain why these existing systems are not meeting the user's needs.

If you are doing an investigatory project (e.g. Big Data – using live data feeds) then you would need to demonstrate that you have done some thorough research on your investigatory area:

- Why I have chosen this area to investigate
- Background technical information
- Other relevant investigations done in this area
- The purpose of my investigation

What information should I be looking for?

Although the information you need to be looking for in your formal investigation will vary from project to project depending on the nature of your project – there are a number of areas you should consider when framing questions for your interview, observation, etc.

Table 2 – Possible information needed for the investigation

Include questions about…	Explanation / example
The organisation/department	Find out some background information about what your end user does or what the organisation or department do. You need to have some idea about the scope of their operations as background information e.g. *"Can you give me some background information about your business?"*
Aims and objectives of the current system	Your end user will normally have some kind of system in place at the moment – so you need some idea of what that system is supposed to deliver
General procedures including time-lines/frequency of events	You need to know how frequently the system is used and how it is used e.g. *"What happens when you receive an order?* Followed by: *"How often are orders made, how many items will be in an order?"*
Users of the current system and/or potential users of a new system	Who actually uses the current system? This will probably include your end user but might also include other users, as well as users of different types. e.g. for a revision system – there will probably be a number of admin users who set quizzes and upload questions but there may be a large number of student users who might use the system to revise
Information/data gathered to be **input** – what and how?	What data items are actually input into the current system?
Documents/reports that are produced using the gathered data, i.e. **outputs**	What kind of outputs might the current system produce – this might be anything from written reports to screen displays, sending emails or messages etc.
How the data is **processed** into the output information	This is about what processing methods are used to transform the input data into the output information. *For the Example in Fig 3 – this may be the calculations that are done to work out how much material is needed to fulfil an order.*
Security, storage of data and backup issues	You do need to ask some questions about how securely they store data (and on what type of storage device if it is done electronically) at the moment particularly if they are subject to the Data Protection Act. Additionally questions that find out how often they back up or archive data will also be useful?

Include questions about…	Explanation / example
Problems encountered during any of the above procedures including errors caused and encountered	This is about what kinds of issues or problems they are having with their current system that has prompted a desire to upgrade / renewal / change
Hardware and software available on-site and/or planned to purchase for the new system	What kind of hardware / Software do they have available and are they willing to purchase new if this is not suitable for the new system. This may well be a limiting factor if they are not.
Users' ICT skills – current ICT usage, training etc.	What level of ICT skills do the prospective users of the new system have? This may well have an impact on your interface design and any training they may require to familiarise themselves with the new system
Client's requirements regarding a new system with reference to the above	What does you client / end user think they need in a new system? Have they got any specific requests with regard to a new system

What evidence of my investigation needs to go in my report?

- You should **include your actual interview / observation notes, copies of documents, copies of questionnaires or evidence of research** in the body of your Analysis or as an appendix to the analysis section.
- **Write a summary of what you found out** from your investigation methods
- **Summarise what you feel are the needs of your project / end user** arising from your investigation and what you found out.

2.2 – Describing the problem to be solved

This is the point at which you start writing the formal report using the information you found out in your formal investigation – starting with a description of the problem you are attempting to solve. This will include:

Background information about the organisation / End-User / Users / Problem / Investigatory area

Describe the organisation (if there is one) in very **general** terms and give some brief background information about it, i.e. nature of its business; is it a large/small organisation. If you have a specified end-user, then identify them. Additionally, it is good to identify any other users of the system. Fig 6 shows an example of how this may be done

Figure 6 – Classic Furnishings: Background Information

Classic Furnishings

Classic furnishings is run by a sole proprietor by the name of Mrs Sandy Smith. She has run this business herself from her own home for more than 10 years. She does not employ any other people. She designs, makes and fits mostly curtains and blinds, although she does not make any other soft furnishing items like bed covers and cushion covers on request. All soft furnishing items are bespoke and made to the individual customer's requirements.

…

If your problem does not have an organisation or if it is an investigatory project, then give some general background information to your project area.

Overview of the problem

This is your opportunity to explain the rationale for the new system or whatever you plan to build. What problems exist with the current system or what gap in the market will your system fill or what gap in the market will your system fill? Make sure you break the problem down into its specific parts. Fig 7 shows an example from the Classic Furnishings scenario.

Figure 7 – Classic Furnishings: Overview of the problem

...

Problems with the current system

Mrs Smith is looking to update her system as she is finding that she is spending too much time doing paperwork. This takes away from the time she needs to spend making and fitting soft furnishings and visiting new customers. Additionally, she would like to be able to give customers a much more speedy quotation for an order. The two aspects of her problem in detail are:

Recording customer personal details

Mrs Smith currently records her customer's personal details in an address book which she normally carries around in her workbag. When she gets home the address book is transferred to her desk. There are various problems with this:

> **Sometimes she leaves it at a client's house – this causes a problem if she needs to use an address from it later at home when making up quotes. Sometimes she loses her address book altogether – this causes a major problem as the only other place she might have addresses is on previous correspondence she has sent to a customer – but it then takes time to find the information that she needs. Mrs Smith also worries about security and an unauthorised person picking up her address book with her customer's personal details in.**

Recording/Calculating orders

When she goes out to visit a customer, she records the measurements of whatever soft furnishings she is being asked to produce. Each new order and its measurements are written on a new page in her order book. Materials required will be added to this order when the customer has chosen them and notified her. Mrs Smith sends an email or letter to her customer outlining the order, the materials required and showing a breakdown of costs including her labour and VAT. This letter is not done until she gets home. There are several issues with this:

> **Mrs Smith is unable to give a precise idea of price until after she leaves the customer, goes home and works it all out. It can be as long as a week before the customer receives the quotation. Quite often, customers decide not to go ahead with the order because they change their mind or she has to go out again to see them because they decide to go for cheaper materials. Either way, this causes loss of profit for Mrs Smith as well as being a very time-consuming exercise.**
>
> **Making up a quotation is a time-consuming process as Mrs Smith has first to look up all of the prices of the different materials from her sample books (she has 15 sample books from different suppliers) and then use a calculator to work out the cost of each material that makes up the order. She then has to calculate a profit margin on the materials, add labour costs, calculate VAT on the whole order and calculate the total.**

Limitations and Constraints imposed

This section needs to contain some idea of what type of limitations or constraints you will need to bear in mind when you start forming your objectives and requirements. For example there is no point designing a complicated interface if your intended users are ICT novices or have particular accessibility needs.

The following are typical constraints and limitations to investigate and comment upon:

- **User's ICT skill level** – this will have an impact on the type and complexity of the interface you design and build
- **User's Accessibility needs** – are any of the potential users visually impaired or have other accessibility needs. This again will impact on the design of the interface that is required.
- **Legal issues concerning Data Protection** (for systems that will collect and store personal data). If your system will potentially store personal data you will need to comply with current legislation covering data protection issues. You should state what principles may cover your system and what constraints this may impose (e.g. the need for security)
- **Hardware and software restrictions** – what kind of hardware and software does the end user have available for you to install the potential new system on
- **Time constraints** – you may need to restrict your project to a particular area of the problem in order to ensure you can complete within the available time frame.

2.3 – Modelling the problem or the existing system

The purpose of this section of evidence is to describe through diagrams and descriptions how the existing system works or aspects of the problem you intend to solve. This can be done through a variety of different diagrams with each type of diagram having a specific purpose. You should ensure that you use the diagrams appropriate for the purpose. So, for example, if your problem area involves data handling then you really need to include Entity Relationship Diagrams and Data Flow Diagrams. On the other hand if you are doing a route-finding program then you will need to include additional items like maps, graphs or trees and possibly adjacency lists or matrices. If you are doing a game then it might be useful to include object diagrams to show what objects (characters, sprites, backgrounds etc.) are used in games of this type and what their likely behaviours are. It is not possible to cover the entire range of diagrams that might be needed but the most commonly needed ones are covered in this section.

Input, Process, Storage, Output (IPSO)

As per the Fig 8 diagram, the purpose of an IPSO is to list what data items are input into the system, what processing actions are carried out on the input data, what data items are then stored in the system and what type of information is output from the system. This gives a good overview of the system or problem area. This is a suitable diagram for all types of system.

Figure 8 – IPSO Overview

INPUT	PROCESSING
What date items are put into or entered into the system	What processing actions are carried out on the data (sorting, searching, calculations)
STORAGE	OUTPUT
What data items are stored in the system	What types of information are output

Table 3 shows an example of an Analysis IPSO chart. This type of layout for an IPSO is particularly useful because it connects the investigation you did with your description of the system by linking in where the evidence for what you are describing can be found in the investigation.

Table 3 – Classic Furnishings: Analysis IPSO chart

IPSO	Information	Evidence
Input	Customer Information: - First Name - Address - Surname - Telephone Number - Email	Observation Notes/Interview
Input	Order Information: - Date order placed - Width of material - Window dimensions (height * width) - Length of required drop - Type of material chosen - Fullness of curtains required	Observation Notes
Process	Calculate amount of material required (drop * width * fullness)	Observation Notes
Process	Calculate cost of making up the curtains (quotation) Amount of material * material cost per metre Total cost of material + fitting + labour + VAT	Observation
Store	Customer information (see items above) Order information (see items above) Quotation information	Observation Notes
Output	- Quotation information- Final invoice	Observation notes/Interview

Data dictionary

The purpose of a data dictionary is to state the data types of the data items used by the system and how these are validated (if at all). Table 4 shows an example of a data dictionary for the classic furnishings scenario. This is a suitable modelling technique for all types of system.

Table 4 – Classic Furnishings: Data Dictionary

Data Item	Data Type	Validation	Sample Data
Customer Firstname	String		Janet
Customer Surname	String		Brookes
Customer Email	String		janetbrookes@gmail.com
Customer Telephone Number	String	= 11 digits	07790034045
Customer Address	String		42 Stoney Lane, Newcastle, ST5 6GB
Date Ordered	Date	>= current date	15/3/14
Material Type	String		Cotton Daisy Print
Material Number	Integer		345456
Window Height (m)	Real		2.2
Window Width (m)	Real		3
Drop required (m)	Real		3.6
Material Width (m)	Real		1.25
Fullness Required	Real		2
Amount of material required (m)	Integer		15

Data volumes

You should provide either an explanation of the volumes of data that flows through the current system (see above) or a table of volumes. This is to give an idea of how much data will be input, output, processed and stored on a daily or weekly or monthly basis. This is a particularly valuable technique to include if you are doing a data-handling problem or a problem that includes some significant data-handling.

Table 5 – Data Volume Table

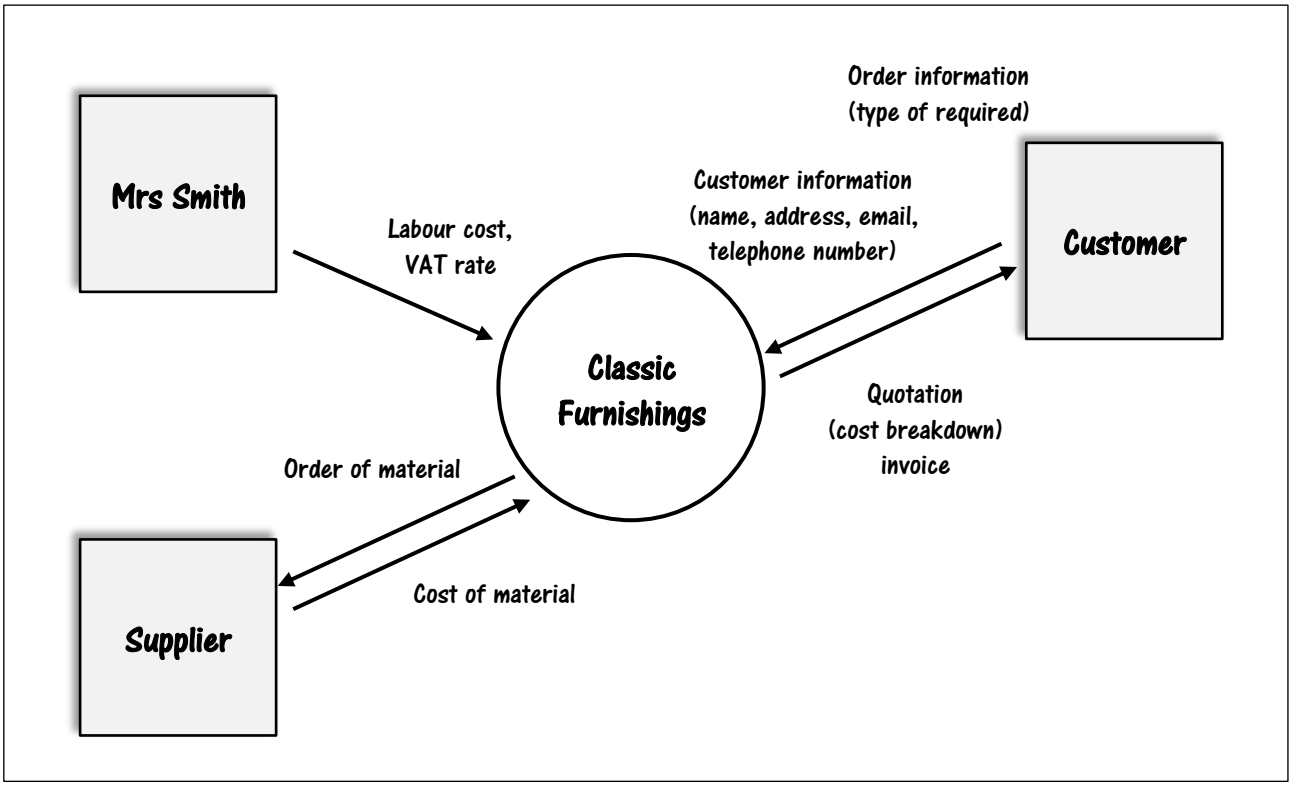
Data object	Volume of data
New customers	Average of 20 per year
Customer quotations given	Average of 136 per year
Customer orders made (and invoices generated)	Average of 110 orders per year
No of items in an order	Average of 4 items per order
Supplier orders generated	Average of 105 orders per year

Context diagrams

A context diagram (sometimes referred to as a level 0 data flow diagram) shows the sources and destinations for data in the system (who or what provides information and who or what does it go to from the system). This is a good technique to use if you are doing a data-handling problem or a problem that includes some significant data-handling.

Fig 9 shows an example of how a context diagram might be displayed. Note that the direction of data flow arrows are annotated to explain what data items flow in each direction.

Figure 9 – Classic Furnishings: Context Diagram



TIP: Drawing Diagrams – online tools

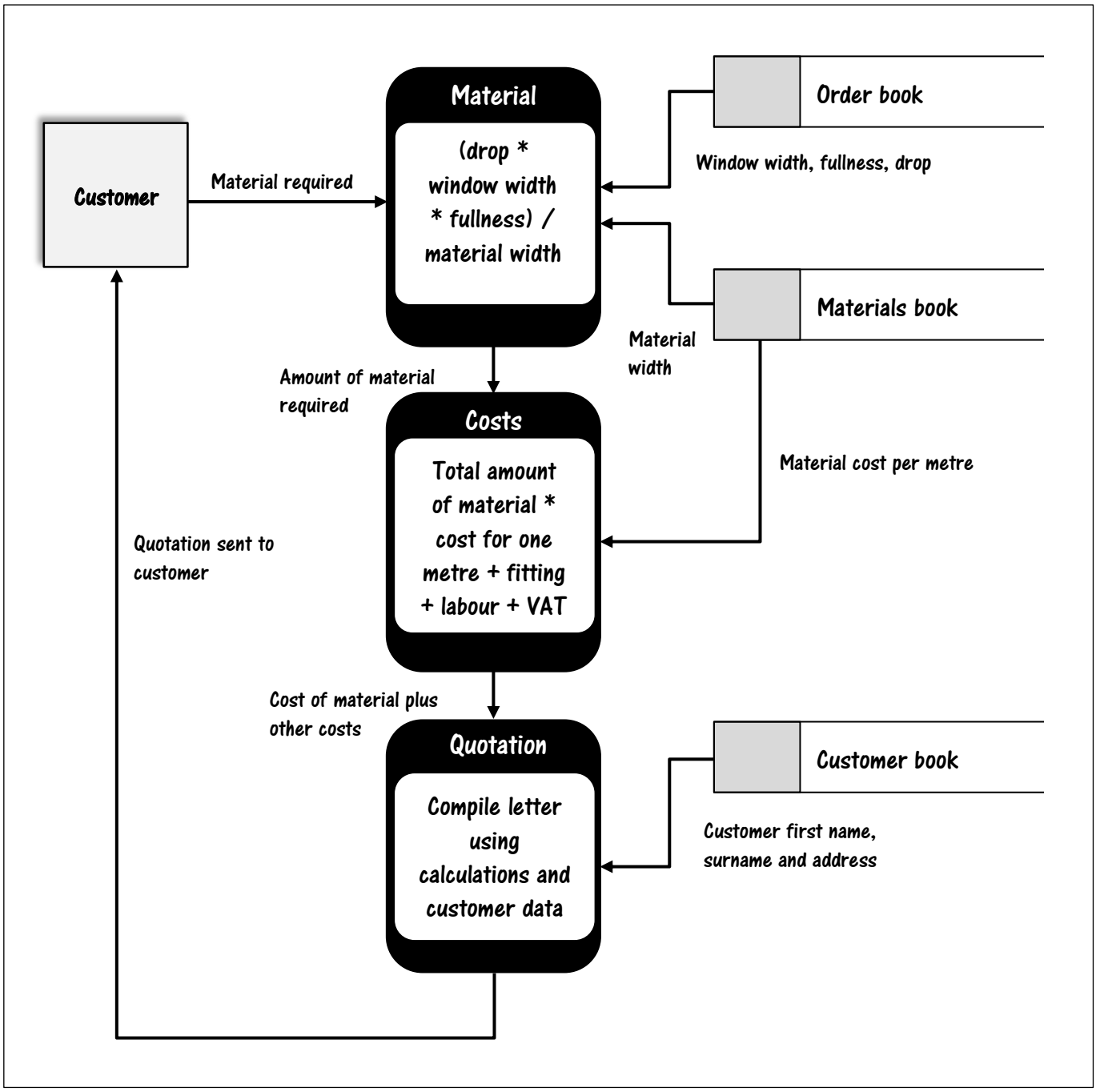
Whilst the tools in various word processing programs can be adapted to draw the various required diagrams you may find it useful to use one of the free programs available. An example of this is **https://www.draw.io/**

DFDs (Data Flow diagrams) and/or Systems Flowcharts

The purpose of a data flow diagram is to show the flow of data through the system. You should provide a Level 1 DFD for each of the major processes that occur in the current system.

Fig 10 below shows an example of a Level 1 DFD for the classic furnishings scenario, showing how a quotation would be produced. DFDSs are particularly appropriate for problems that include significant data-handling whilst flowcharts are generally suitable for all types of problem.

Figure 10 – Classic Furnishings: Example Level 1 DFD
(Showing the process of compiling a quotation)

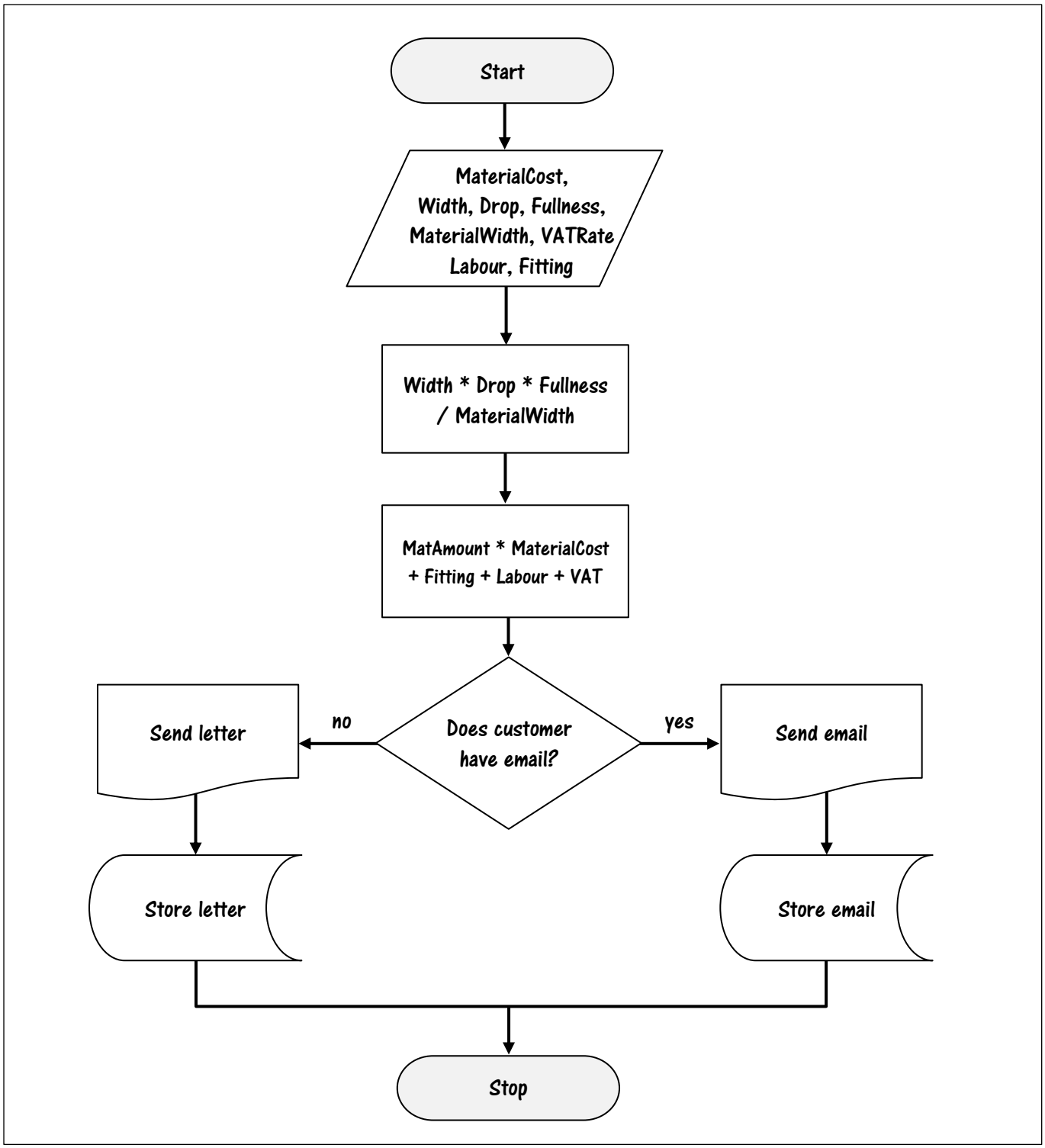


Systems Flowcharts

Sometimes it is more appropriate to use a Systems Flowchart to show how a system works and this is particularly true of non-data-handling projects. Whilst a DFD shows the flow of data through a system a systems flowchart shows the order of input, output, processing, and storage events and also allows the display of branching events (where decisions are made and one of two different events occurs).

Fig 11 below shows how a flowchart might be compiled for the generation of a quotation for a customer in the classic furnishings scenario.

Figure 11 – Classic Furnishings: Example Flowchart
(Showing the process of compiling a quotation)



ERDs and EAM

An **E**ntity **R**elationship **D**iagram (see below) is a diagram that shows the degree of relationship between the entities in a system. An **E**ntity **A**ttribute **M**odel also shows the degree of the relationship between entities but additionally shows what attributes each entity has. An Entity Relationship Diagram and Entity Attribute model are needed if you are working on a data handling problem. Complete an Entity Relationship Diagram just for the entities and then expand it to an Entity Attribute Model showing what attributes are stored for each entity.

Terminology

Entity – an object person of thing about which data is to be held
Attribute – a characteristic of an entity

The Fig 12 shows an Entity Relationship Diagram and Fig 13 that you can see the corresponding Entity Attribute Model.

Figure 12 – Classic Furnishings: Example Entity Relationship Model

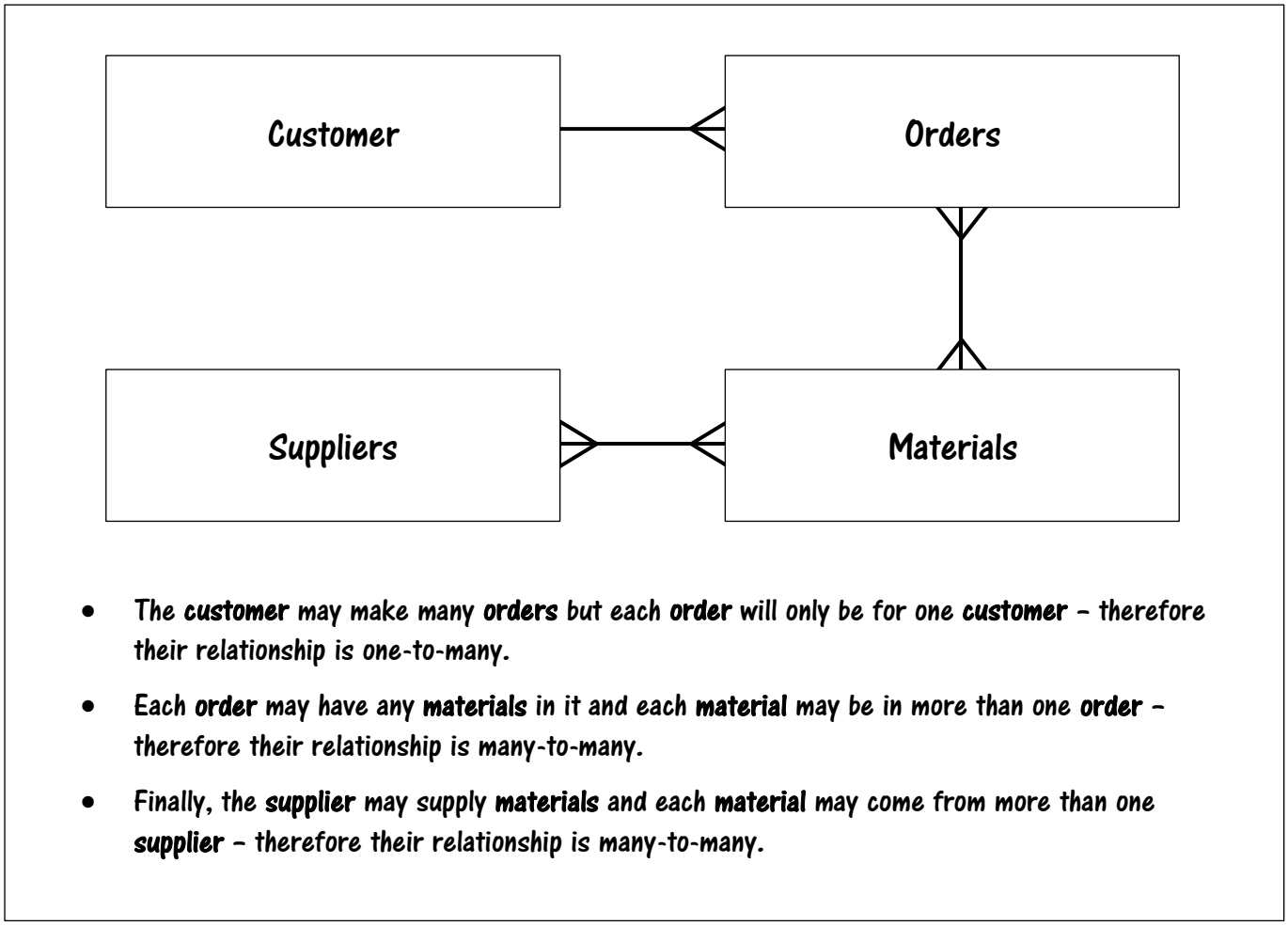
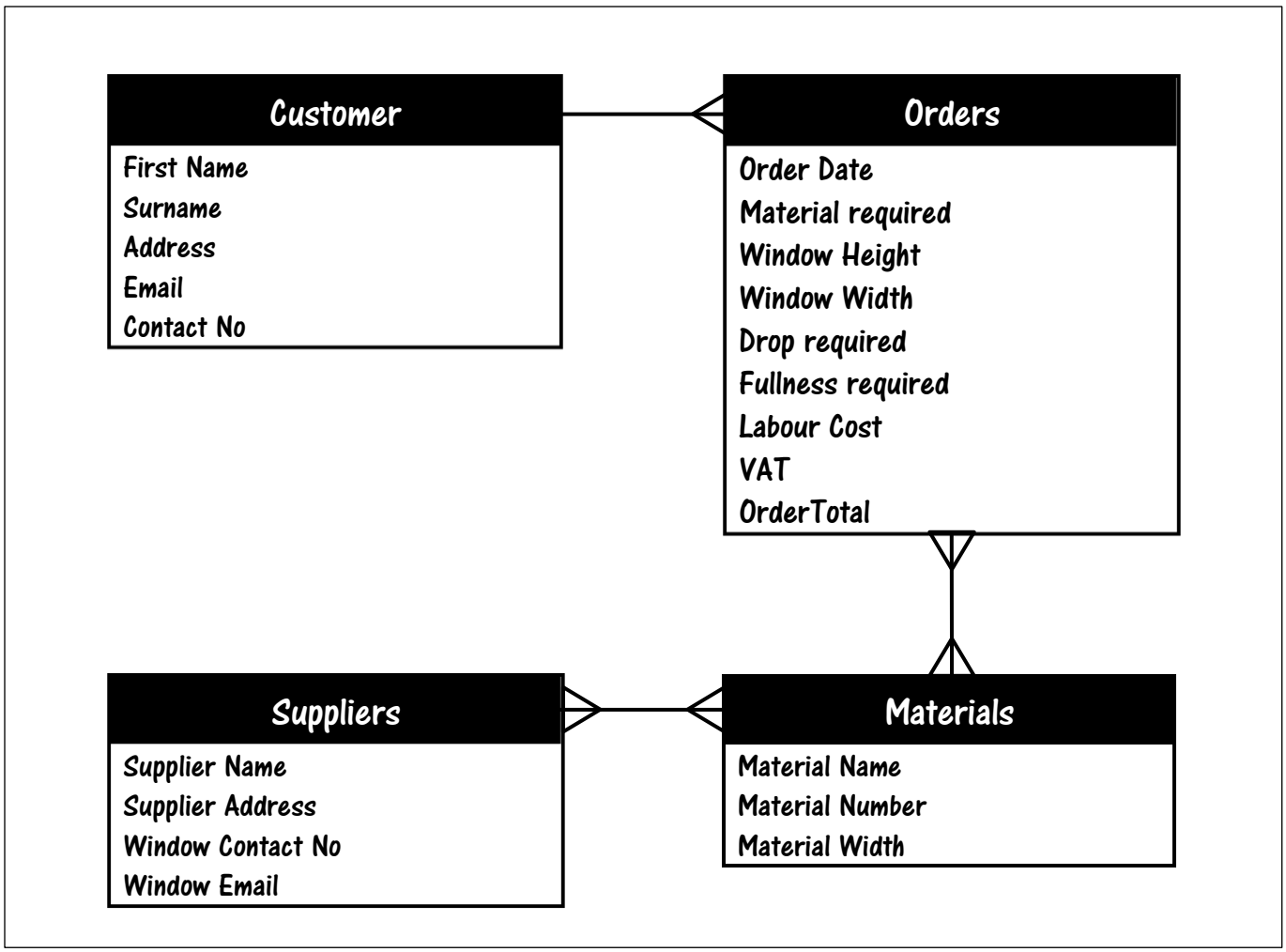


Figure 11 – Classic Furnishings: Example Entity Attribute Model



2.4 – Describing the specific objectives of the new system

(including performance criteria)

It is important in this section of your analysis that your objectives indicate that your project is **of A Level standard**. You can do this by making sure that your objectives refer to A-Level concepts that might be used to implement an objective. For example, using a queue data structure is an A-Level concept. Therefore an objective could be framed that included a reference to a queue being used.

Another example from the Classic furnishings scenario is:
I will create a secure storage area for customer details and their orders.

It could be made clear that this was an A-Level objective by adding:
I will create a secure storage area for customer details and their orders using a database. It will be secured using hashed passwords for users with access.

- **Purpose of this section**
  The purpose of this section of the analysis is to state what the objectives of the system are and how their success will be judged. Your objectives will arise from two possible sources:
  - As a consequence of the identification of a specific problem (arising from your investigation and dialogue with an end-user or potential users)
  - As a consequence of a direct request from your end-user
- **Definitions of key terms**
  - **Objectives** – What the system must be able to do when complete
  - **Performance criteria** – how will you know that a particular objective has been successful?
  - **Qualitative objectives** – are objectives that can't easily be measured and are very often subjective (e.g. the system should be user friendly) but are important for the project to be successful
  - **Quantitative objectives** – where the successful outcome can easily be objectively measured

Setting out objectives and performance criteria

- **Make a numbered list of your objectives**
  Using a table similar to the example shown in **table 6** – you need to describe your objectives for the new system based on what you found out about your user needs during your investigation. You should try to add performance criteria so that your objectives are measurable (you know how the success of your objectives needs to be judged).

Table 6 – A set of objectives and performance criteria for some of the Classic furnishings scenario

No	Objective	Performance Criteria
1	Create a secure storage area for customer details using a database and secured with hashed passwords for users with access	This must be secured so that only the end user can access this data.
2	Allow customer data to be searched, updated, archived and inserted using SQL	It must be searchable by customer surname or first name to find customer phone numbers so that the client can arrange appointments to visit or fit curtains. It must be searchable by customer surname or first name so that the client can find addresses when she needs to visit. Additionally, she must be able to update specific customer details when they change houses or phone numbers etc. Must be able to archive old customers when no new orders have been received for 5 years.

No	Objective	Performance Criteria
3	Create a storage area for materials and their suppliers and allow this data to be searched so that the client can get prices when compiling a quote and send material orders off to suppliers so that she has the materials to fill an order	These need to be searchable by price range, colour, type of material, material name or supplier. Searching for a supplier will also bring up the supplier address and email so that a material order can be sent off for materials to make up the customer order.
4	Create a feature that allows the client to input window dimensions and work out how many metres of material will be needed	It will work out the number of metres of curtain fabric (and possibly trimmings if selected) required based on the input dimensions of the window. These should be stored and used later in the generation of the quotation and invoice
5	Create a feature that allows the client to add materials to an order and works out the total cost of the potential order including labour and VAT to generate a quotation	It will allow the client to select the materials to be used for a potential order. It will work out a quotation based on the: • Number of metres of material needed • The cost of the materials selected per metre x number of metres needed (or number of items selected) • Add a labour cost based on the number of hours the client thinks it will take her to complete the order • Calculate a sub-total • Add VAT based on the sub-total • Calculate the final total cost including Vat When all costs have been calculated it should print out or email a quotation for the customer. This information should be saved in case the client needs to amend a quotation.

2.5 – Consideration of possible solutions to the problem

Programming languages, developer environments and platforms)

This section involves some consideration of what types of programming language, environments and platforms might be used to build the new system. While it generally holds true that most students will choose to use the programming language they have been taught at their school /college there does need to be more consideration than simply choosing a language because it's the only one you know how to program in.

NOTE – be careful of choosing developer environments that code for you. The emphasis of this project is on demonstrating your coding ability. For example, some programming environments (through the use of wizards) allow you to connect to and interrogate a database without much in the way of actual coding. However, to demonstrate coding ability it would be expected that a student would write the code themselves that connects to a database, creates tables, datasets and commands that search the database. The same is true of app development environments that use drag and drop components with limited use of coding.

DATA-HANDLING PROJECTS – whilst it is acceptable to use database software to store data in relational data storage areas (tables) – you should do the following in code:

- Consider using CREATE DB & CREATE TABLE statements to create databases, tables / relationships when the system is run for the first time
- Use code to create objects like connections, datasets, data tables and data commands.
- Create queries using SQL statements (embedded in code)

Analysis: Evidence Checklist

Key to the importance of sections of evidence:

- Any item marked with an **E is essential** and **must be included** if you are to gain good marks in this section
- Any item marked with a **D is desirable** and **should be included,** it will help to gain good marks but is not always essential
- Any item marked with a **C is complementary**. It may help explain aspects of your project but is not essential

Item No	Have you included???	Tick
1	**Evidence of an investigation** (interview notes, questionnaires, online forms, research notes, observation or examining documents notes) This should include: • **Actual interview / observation notes, copies of documents, copies of questionnaires or evidence of research** in the body of your Analysis or as an appendix to the analysis section. **[E]** • **Write a summary of what you found out** from your investigation methods **[E]** • **Summarise what you feel are the needs of your end user/ project** arising from your investigation. **[E]**	☐
2	Describe the **background** to your problem / project **[E]** and identify your **end user (if you have one) [E]**	☐
3	**Describe the problems** faced by the existing system in detail (if there is one) **[D]**	☐
4	Describe the **constraints and limitations** that any system will need to contend with in this organisation **[C]**	☐
5	**Model the existing system or project problem** in detail using appropriate **diagrams and modelling techniques where appropriate** (IPSO chart, Analysis data dictionary, Context diagram, DFDs and/or systems flowcharts, ERDs and Entity Attribute models, Data volumes) **[E]**	☐
6	**List your objectives** (including performance criteria for each one) **[E]**	☐
7	**Consider the potential solutions** to the problem (languages, developer environments and platforms) **[C]**	☐

Analysis: Assessment Criteria

There is a maximum of **9 marks** awarded for this section

Mark band	Advice	Marks
Upper Mark band	The real problem / investigation you are intending to solve has been fully investigated and described which means that somebody else can easily understand the scope of the problem(s) you are intending to solve It is evident that your objectives cover all the needs of your end user as described in your analysis documentation. There is clear evidence of dialogue with end user / potential users that have informed the objectives. All your objectives should be fully documented and cover all the required functionality for a problem / investigation of this type**. They should be measurable (i.e. you have set performance criteria and will be able to prove that they have been achieved) and are appropriate for the problem / investigation you have described. Your description of the existing system (using modelling techniques) is thorough enough that it will be of use in the later stages of your project like design.)	7–9
Middle Mark band	The real problem you are intending to solve has been investigated and described although there are some omissions that don't prevent the scope of the problem(s) you are intending to solve being understood by someone else It is evident that your objectives cover most of the needs of your end user as described in your analysis documentation. There is some evidence of dialogue with end user / potential users that have informed the objectives. Most, but not all, of your objectives are fully documented – they are measurable (i.e. you will be able to prove that they have been achieved) and appropriate for the problem / investigation you have described** Your description of the existing system (thorough modelling diagrams) may have some omissions but will still be of reasonable use in design and other later stages. For example, a data-handling solution that has no entity-relationship modelling or a route-finding problem that has no maps or graphs / trees.	4–6
Lower Mark band	The problem you are intending to solve has been partially investigated and described It is evident that there is some attempt through dialogue to discover the needs of your end user or potential users. Your objectives are partially documented – they are not all measurable and / or appropriate for the problem / investigation you have described**. They cover only some of the required functionality for a problem / investigation of this type. Your description of the existing system (thorough modelling techniques) will be of some use in design.	0–3

** Whether your objectives cover all the required functionality of your problem or investigation will be of crucial importance in the technical solution. You are awarded marks there (out of 15) based on how complete your solution is. This is not just about how many of your original objectives you have completed but is also about what objectives you could reasonably be expected to have for a problem or investigation of the type identified in your Analysis.

For example, a student has completed a solution for a quiz program to help students revise for a subject. The student achieves all the original objectives set in Analysis but failed to include an objective that would allow the teacher to add new questions to the program. This solution could not then be awarded full marks for the solution (even though all the original objectives had been achieved) as it is reasonable to expect a revision system to include a facility for updating/ replacing/ adding new questions – otherwise such a system becomes redundant pretty quickly.

3. Getting Started with Programming

It is a good idea (once you have a set of initial objectives) to make a start on the coding as early as possible, particularly if your solution is going to involve techniques you have never used before and therefore need to be researched. Before you start any coding you should choose the critical path through your project.

3.1 – Choosing a critical path through your project

This refers to the order in which you develop your project objectives. This will be different for everybody but to start with you make a decision about which objectives are **critical to the success of your system** and start with those. For example, if I am developing a game to quiz KS3 students about computing topics then the critical path will be the part of the program that asks questions and checks answers. Any log in system, admin rights, scoring system, leader board and interactive graphics can be added in later prototypes. See example 1 below:

Example 1

A student has been asked by a school to develop an entertaining game to test student's space science knowledge.

As Space is one of the topics that these Year 6 students study the school wants the game set around an entertaining space theme. The student has started their project with these initial objectives:

1	The game must allow two players to play against each other
2	When the game starts the program works out a space name for each player based on extracting substrings from the player's name, place of birth, surname and mother's maiden name.
3	The program must allow each player to place their spaceships on their playing board
4	The program must draw actual spaceship pictures using drawing tools
5	The program must check that each player has the required number and type of spaceships
6	The program must randomise who goes first
7	The program must allow each player in turn to choose a location on the grid to fire a missile at: • The location choice must be checked to ensure it is valid • The program must ask the player a space science question • If they get it right then their chosen location is fired at. If they get it wrong then the other player gets their turn • A message should be output (hit, miss or destroyed – depending on whether there is an opponent's ship in that location) • If all spaceships for the other player have been destroyed then a game won message should be output
8	The program should produce output a list of the last 5 (space names) of players who have recently won games

The critical path through these objectives is to attempt 1, 3, 6 and 7 first and produce a partial solution that fulfils those objectives before then going on to tackle additional objectives. In this example the second critical path might include objectives 2, 5 and 8 and, when those are complete, any remaining objectives to complete the solution. In this way those objectives that are critical to the success of the solution are tackled and achieved early on in the development of the system.

3.2 – Prototyping

A **prototype (in programming terms)** is an example of your system which will not be fully functional. It will have a few to some of the capabilities of the intended final product. Building a prototype (or series of prototypes) is useful because it allows you to involve your end user in the development process

As each prototype is built it can be used to tease out further requirements from your client and get valuable feedback which helps further development. It enables your end user to see what the system will look like and how it can be used.

You can choose this as a method to develop and document your project and if you do then your analysis design, technical solution, testing and evaluation becomes more of an iterative process. If you do choose this method then make sure you have easy access to your end user as their feedback is vital and could hold up your progress if they are difficult to get hold of.

How many prototypes should I build?
This is a difficult one to state a definitive answer to, as all projects are different, but in general around 1 or 2 prototypes and a final product would probably be enough for most A-Level projects.

How would I document my project if I used prototypes?
If you do use prototype versions then it is a good idea to reflect that in your documentation. The table below (Table 1) shows how you might document a prototype solution

Table 1

Sections	Prototype 1	Prototype 2	Final product
Analysis	**Do your analysis section as normal – for the whole problem**	**Review the objectives** – do any need amending, removing or adding in light of the feedback received from Prototype 1? Choose the **next critical path** through the remaining objectives	**Review the objectives** – do any need amending, removing or adding in light of the feedback received from Prototype 1?
Design	**Design Prototype1**- cover all the required areas for the objectives you plan to build in your first prototype. This first prototype should include the critical areas of your project (see previous chapter on choosing a critical path)	**Design Prototype 2**- you need to cover the required areas for the next set of (critical path) objectives you plan to add to the system. , e.g. *it might just involve some extra interface design and additional algorithms but might not need any further alterations to the data dictionary or other design items.*	**Design the final product**- Cover the required areas for the remaining objectives you need to satisfy in order to have a fully working solution
Technical Solution	**Build and document** prototype 1	**Build and document** the additional areas of prototype 2	**Build and document** the additional required areas
Testing	**Test plan and testing screenshots** should cover the built objectives	**Test plan and testing screenshots** should cover the additional built objectives	**Test plan and testing screenshots** should cover the additional built objectives and any testing needed for the whole system
Evaluation	**With your end user** evaluate the success of this first prototype. Get some feedback from them and revisit your objectives in light of this feedback	**With your end user** evaluate the success of this second prototype. Get some feedback from them and revisit your objectives in light of this feedback	**With your end user** evaluate the success of the final product. Get some feedback from them and suggest further possible extension in light of the feedback they give

4. Documented Design Section

In this section of your project you are designing a solution to the objectives and requirements that you identified in your Analysis section and this must be evident in your design documentation. A relatively competent 3rd party should be able to implement your ideas from your Analysis and Design documentation.

Whilst a top mark band design will include all elements of design the most important section of design is the processing design (algorithms, data, data structures) and for a data-handling project, data structure and extraction design plus algorithms.

4.1 – Documentation advice specific to data-handling projects

If your project will involve any data handling using a relational data structure (two or more related tables) then you must provide evidence of:

1.1 An **entity relationship diagram** (ERD) and an **entity attribute model** (EAM). These may be similar to the diagrams you used in Analysis although there may be subtle differences (more or less entities and/or more or less attributes for those entities)

1.2 Putting the attributes identified into **standard notation for database tables** and **normalising** to at least 3rd normal form and possibly Boyce-Codd.

1.3 Producing **a post-normalisation Entity Relationship diagram**. This will be slightly different to the pre-normalisation diagram as extra (link) tables may have been added.

1.4 Producing **data dictionaries for each table** based on the normalised design

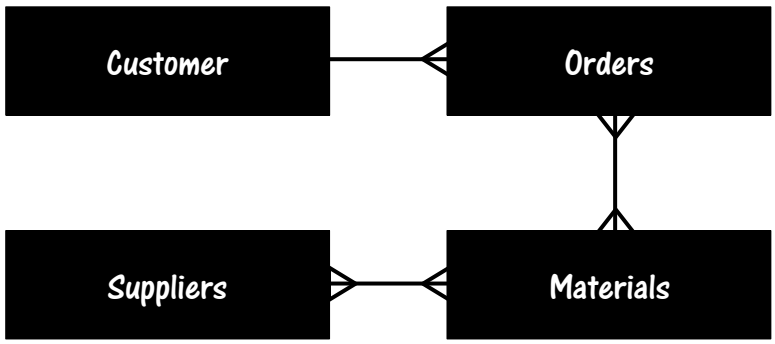
1.5 Producing **SQL statements** that will be used in algorithms to create the database and described tables, extract, import, update, delete and archive data – to meet the identified data requirements of the system.

1.6 Producing **Data Flow diagrams** for the main data processing objectives

Entity Relationship Diagrams and Entity Attribute Models

The following diagrams are examples of how to present your evidence. First draw your ERD. You can do this by copying your ERD from Analysis and making any subtle changes that are necessary to reflect your requirements. **Fig 14** represents the ERD for the Classic Furnishings scenario before normalisation.

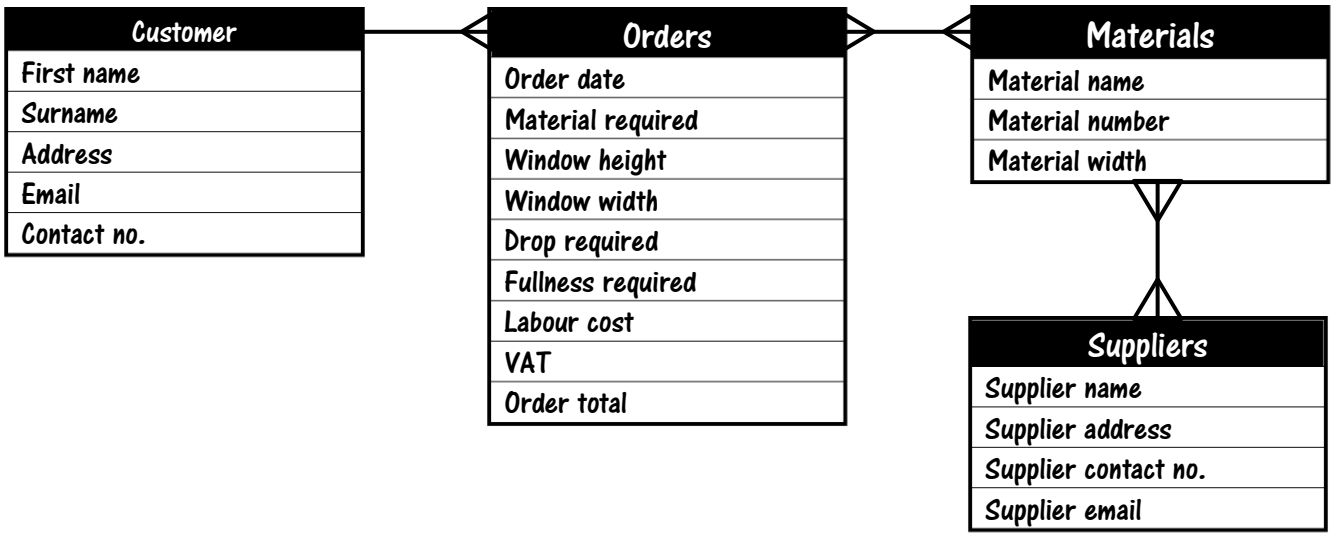
Figure 14 – Classic Furnishings: Example of a Pre-Normalisation Entity Relationship Diagram



- The **customer** may make many **orders** but each **order** will only be for one **customer** – therefore their relationship is one-to-many.
- Each **order** may have any **materials** in it and each **material** may be in more than one **order** – therefore their relationship is many-to-many.
- Finally, the **supplier** may supply **materials** and each **material** may come from more than one **supplier** – therefore their relationship is many-to-many.

Once you have done this then expand the diagram to include the attributes that need to be stored about each entity. **Fig 15** shows the EAM for **Fig 14**.

Figure 15 – Classic Furnishings: Example of an Entity Attribute Model



Normalising the Data structure?

Once you know what attributes need to be stored about each entity then they can be placed into standard notation for database tables. **The following example** shows the EAM above placed into standard notation. Note that the primary key for each table is placed as the first attribute inside the brackets.

Tables / Fields for Classic Furnishings Data Structure

tblCustomer(**CustEmail**, FirstName, Surname, Address, Town/City, PostCode, ContactNo)
tblOrders(**OrderID**, OrderDate, LabourCost, VAT, OrderTotal, WindowHeight, WindowWidth, DropRequired, FullnessRequired, MaterialName)
tblMaterials(**MaterialNumber**, MaterialName, MaterialWidth, CostPerMetre)
tblSuppliers(**SuppEmail**, SuppName, SuppAddress, SuppTown/City, SuppPostCode, SuppcontactNo)

A thorough design will show the stages of normalisation as well as a normalised design

First Normal Form for Classic Furnishings (RA) stands for repeating attribute

tblCustomer(**CustEmail**, FirstName, Surname, Address, Town/City, PostCode, ContactNo)
tblOrders(**OrderID, MaterialNumber (RA)** OrderDate, LabourCost, VAT, OrderTotal, WindowHeight, WindowWidth, DropRequired, FullnessRequired, MaterialName, **CustEmail,)**
tblMaterials(**MaterialNumber**, **SuppEmail,** MaterialName, MaterialWidth, CostPerMetre)
tblSuppliers(**SuppEmail**, SuppName, SuppAddress, SuppTown/City, SuppPostCode, SuppcontactNo)

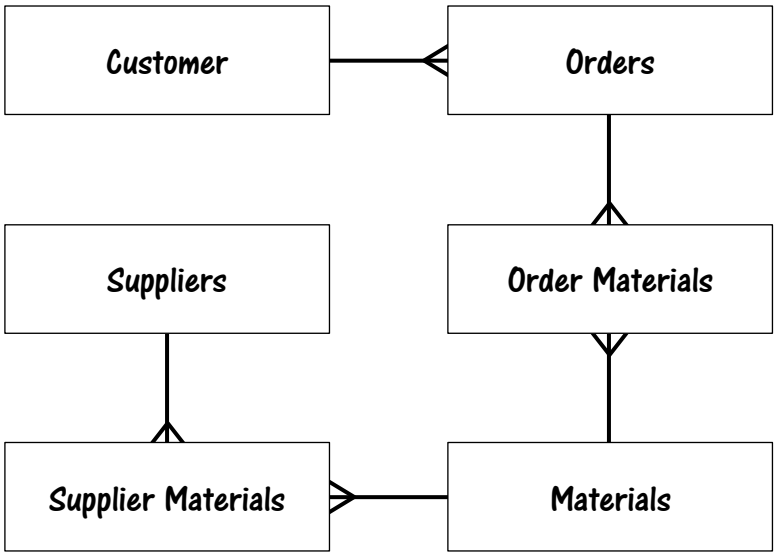
Second Normal Form for Classic Furnishings (now fully normalised)

tblCustomer(**CustEmail**, FirstName, Surname, Address, Town/City, PostCode, ContactNo)
tblOrders(**OrderID,** OrderDate, LabourCost, VAT, OrderTotal, WindowHeight, WindowWidth, DropRequired, FullnessRequired, **CustEmail,)**
tblOrderMaterials(OrderID, MaterialNumber, NoOfMetres)
tblMaterials(**MaterialNumber**, MaterialName, MaterialWidth, CostPerMetre)
tblSuppMaterials(SuppEmail, MaterialNumber)
tblSuppliers(**SuppEmail**, SuppName, SuppAddress, SuppTown/City, SuppPostCode, SuppcontactNo)

Post-Normalisation Entity Relationship Diagram

Once normalisation is complete then it is good practice to display the normalised Entity Relationship Diagram. This diagram will also help you to set your relationships accurately during the creation of your solution. Below is the normalised diagram for *Classic Furnishings* which clearly shows the insertion of two new tables which indicates that the design now has no remaining many-to-many relationships. Fig 16 shows the post normalisation ERD for Fig 14.

Figure 16 – Classic Furnishings: Example of a Post-Normalisation Entity-Relationship Diagram



Creating Data Dictionaries for each table

In this section it is necessary to create a data dictionary for each table in the data structure that displays **data types** for each field, identification of the **keys** and any **validation** that will be required on the fields at the point of data entry and storage. The examples (Table7 and 8) show how this evidence might be set out.

Table 7 – Classic Furnishings: Data Dictionary for Relational Data Table

Table Name	TBLCUSTOMER		
Primary Key	CustEmail		
Foreign Keys	N/A		
Data Item	**Data type**	**Validation**	**Sample data**
CustEmail	String	NotNull	janetbrookes@gmail.com
FirstName	String		Janet
Surname	String	NotNull	Brookes
ContactNo	String	=11 digits	07790034045
Address	String		42 Stoney Lane
Town/City	String		Newcastle
PostCode	String		St5 6GB

Table 8 – Classic Furnishings: Data Dictionary for Relational Data Table

Table Name	TBLORDERS		
Primary Key	OrderID		
Foreign Keys	N/A		
Data Item	**Data Type**	**Validation**	**Sample data**
OrderID	Integer	>0	125
DateOrdered	Date	>=currentdate	15/3/14
LabourCost	Real	To 2 decimal places – formatted as currency	£50
VAT	Real	To 2 decimal places – formatted as currency Calculated (material cost + Labour *VAT rate	£23.45
OrderTotal	Real	To 2 decimal places – formatted as currency Calculated (material cost + Labour *VAT rate	£480
WindowHeight	Real		2.2
WindowWidth	Real		3
DropRequired	Real	Cannot be less than window height	3.6
FullnessRequired	Real		2
CustEmail	String	NotNull	janetbrookes@gmail.com

SQL Statements

In this section you should list and describe the SQL statements that you will need in order to meet your requirements. For example, your requirements may need you to:

- Create database (once only)
- Create data tables (once only)
- Extract data from data tables
- Insert new records into data tables
- Update existing records in data tables
- Delete or archive existing data

The following examples show how to present this evidence. Remember that for you to score high marks in your technical solution you will need to be running queries that extract, update, delete or insert data from more than one table at the same time.

Example 1 –Cross-table Parameterised Select Query

A query to extract all orders for a particular customer with a particular order date. Results need to show the dimensions of the window for which curtains are to be made up. The customer name and order date to be searched for will be passed as parameters to the search at runtime

```
SELECT FirstName, Surname,
WindowHeight, WindowWidth,
DropRequired, FullnessRequired
FROM Customers INNERJOIN Orders ON
Customers.CustEmail = Orders.CustEmail
WHERE Surname = [parameter] AND
OrderDate = >[parameter]
```

Example 2 –SQL Statement to Create a Database Table

A DDL (Data Definition Language) statement to create the customer table at first runtime.

```
CREATE TABLE Customers
(CustEmail VARCHAR(100) NOT NULL
PRIMARY KEY,
FirstName VARCHAR(50),
Surname VARCHAR(50) NOT NULL,
Address VARCHAR(200),
Town/City VARCHAR(30),
PostCode VARCHAR(8),
ContactNo VARCHAR(11);)
```

Data Flow Diagrams

Try and include a DFD to Level 1 for each of your main data processing objectives.

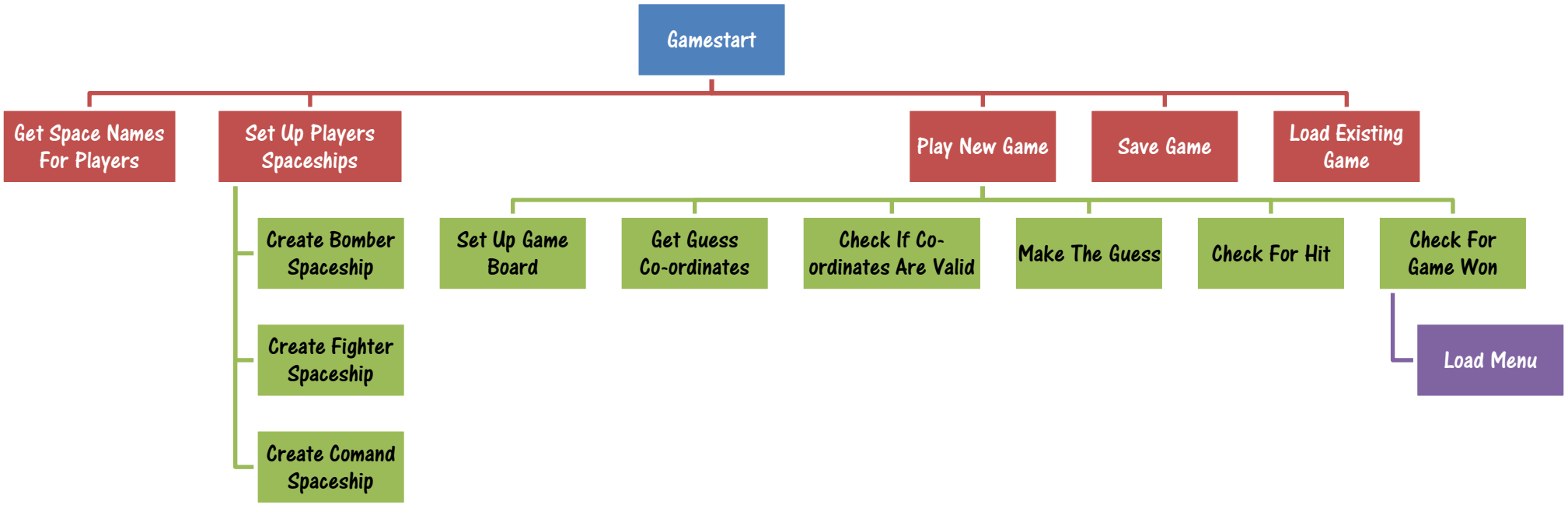
4.2 – Overall system design

The purpose of this section of evidence is to describe the overall system design needed to meet your requirements. This can be done through a top-down diagram and an IPSO chart.

Top-down diagram

The example (Fig 17) below is a top-down design diagram for a program called Space Battleships where up to two players will try to guess where each other's spaceships are in order to win the game.

Figure 17 – Space Battleships: Example of a Top-Down Design Diagram



IPSO chart (Input, Process, Storage, Output)

The purpose of an IPSO is to list what data items are input into the system, what processing actions are carried out on the input data, what data items are then stored in the system and what type of information is output from the system. The IPSO chart in **Table 9** shows how the Spaceships game might be categorised into input, process, storage and output.

Table 9 – Spaceships: An Example of a Design IPSO Chart

IPSO	Program Section	Item
Input	Getting Space Names	Player One First Name Player One Surname Player One Mother's Maiden Name Player One's place of birth Player Two First Name Player Two Surname Player Two Mother's Maiden Name Player Two's place of birth
	Setting up Spaceships	Command spaceship Co-ordinates (X and Y) Bomber spaceship 1 Co-ordinates (X and Y) Bomber spaceship 2 Co-ordinates (X and Y) Bomber spaceship 3 Co-ordinates (X and Y) Fighter spaceship 1 Co-ordinates (X and Y) Fighter spaceship 2 Co-ordinates (X and Y) Fighter spaceship 3 Co-ordinates (X and Y) Fighter spaceship 4 Co-ordinates (X and Y) Fighter spaceship 5 Co-ordinates (X and Y) Fighter spaceship 6 Co-ordinates (X and Y)
	Choosing a game	MenuNumberOption
	Guessing a ship position	Guess Position X Co-ordinate Guess Position Y Co-ordinate
Processing	Setting Up Spaceship positions	Is Grid Position (XY Co-ordinate) empty?
	Check if Valid Guess	Is the Guess a valid XY co-ordinate on the grid
	Check for hit	Do the guess co-ordinates have a ship occupying the space
	Make the guess	If position guessed has a ship, then mark position as a hit
	Check if Game won	Check both player1 grid and player 2 grid – if all ship positions are hits then game is won
Storage	Save Gameboard	All positions on player 1 and player 2 gameboards saved to a text file
Output	Setting Up Spaceship positions	Gameboard for player 1 and 2
	Check if Valid Guess	Message "invalid co-ordinates – please re-enter"
	Check for hit	Message "that is a hit" or "that is a miss"
	Check if Game won	Message "Player (x) has won the game"

4.3 – Data dictionaries / record structure / data structures

Even if you have a non-data handling project you will still need a **data dictionary** that covers the data items used by your program. This should show both elementary and composite data types like structures.

Additionally, if you plan to store records in a text file or DAT file then you need to show your record structure.

Data dictionary

The purpose of a data dictionary is to state the data types of the data items used by the system and how these are validated (if at all). The Data dictionary in **Table 10** shows some of the data items that might be used by the Spaceship program. Where user defined data structures are to be used it should be explained what elementary or composite data types the structures are composed of.

Table 10 – Spaceships: Example of a Data Dictionary (non-database project)

Data item	Data Type	Validation	Sample data
Player first name	string		"Robert"
Player Surname	string	<>""	"Brock"
Mother's maiden name	string		"Wilson"
Place of birth	string	<>""	"Tetbury"
Command spaceship co-ordinates	Command structure (CPos 1, 2 ,3, 4 as spaceship)		(3, 4) (3, 5) (4, 4) (4, 5)
Bomber spaceship co-ordinates	Bomber structure (BPos 1, 2 as spaceship)		(1, 2) (2, 2)
Fighter spaceship co-ordinates	Spaceship structure (X Co-ord as integer, Y Co-rd as integer)	YCo-rd >= 0 AND <=11 X Co-rd >=0 AND <=9	4, 9
Menu option	integer	>=1 AND <=6	6 (Quit)
Guess X Co-ordinate	integer	X Co-rd >=0 AND <=9	
Guess Y Co-ordinate	integer	YCo-rd >= 0 AND <=11	
Player 1 ships	2-D string array		("","","","B","B", "", ""……..)
Player 2 ships	2-D string array		("", "S", "", "", "B", "B"……..)

Record structure

If you are going to be saving records to a text file, then it is necessary to design the structure for one record that will be saved into the file.

In the case of the spaceship program it will be possible to save the state of the gameplay – the example in **table 11** shows how a single record that might be saved to the "Saved Games" file would be constructed

Table 11 – Spaceships: Example of a Record Structure

Record element	Size
Player 1 ships	120 characters = 960 bytes
Player 2 ships	120 characters = 960 bytes
One record = 1920 bytes	

Data structures

If you plan to use a structure, a tree, a graph, stack, queue or linked list then you should show how these will be implemented.

```
STRUCTURE FIGHTER
  FighterName as String
  FighterXC-ord as integer
  FighterYCo-ord as integer
  FighterHealth as Integer
  FighterSize as integer
  FighterDestroyed  as Boolean
END STRUCTURE
```

This example shows the data items and their data types for a User-defined Fighter Structure

LINEAR Queue

At the end of the game the user will be presented with a list of the ships destroyed in the order in which they were destroyed. I will use a linear queue to hold this information. As each ship is destroyed it will be added to their "Destroyed" Queue. When the game is over the Queue will be output to the interface in the order in which items were added.

Player1Destroyed Queue (Array of Records)

[0]	Fighter object
[1]	
[2]	
[3]	
[4]	
[5]	
[6]	
[7]	
[8]	
[9]	

Variables will be needed for FrontPointer and RearPointer (integers) and QueueLength (integer)

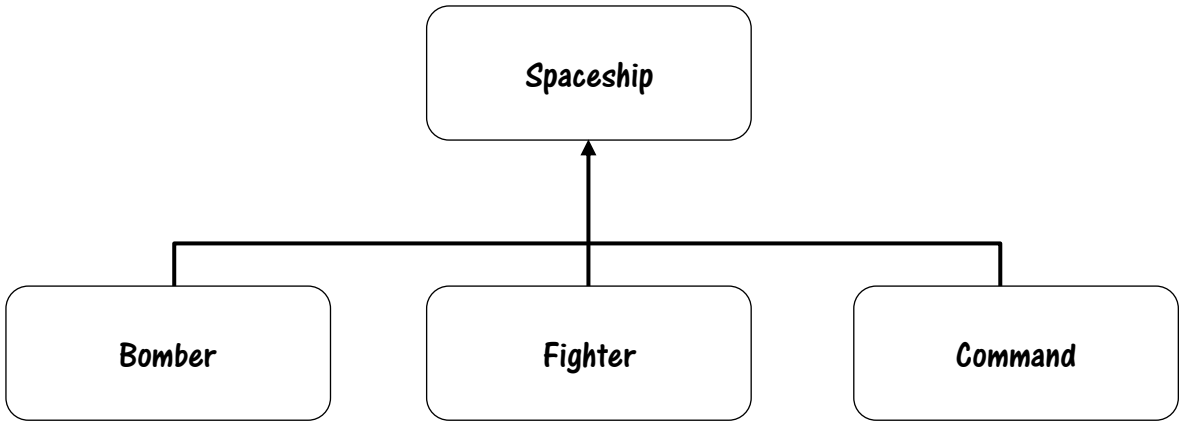
This example shows how you describe the way in which a queue might be implemented

4.4 – OOP class design

If you are going to be using user-defined classes in your program, then you should include both an inheritance diagram and class definitions for each parent and sub class in your design section. When it comes to the marking of your technical solution, using your own defined classes is considered a higher level skill and will help to place you in the higher mark bands.

Fig 18 shows an inheritance diagram for the Spaceship program and class definitions for the parent class and one of the sub classes.

Figure 18 – Spaceships: Inheritance Diagram and Class Definitions




```
CLASS Spaceship
Public Function GetSpaceshipType
Public Sub SetCo-ordinates
-----------------------------------
Private Type : String
Private XCo-rdinate : integer
Private YCo-ordinate: integer
```

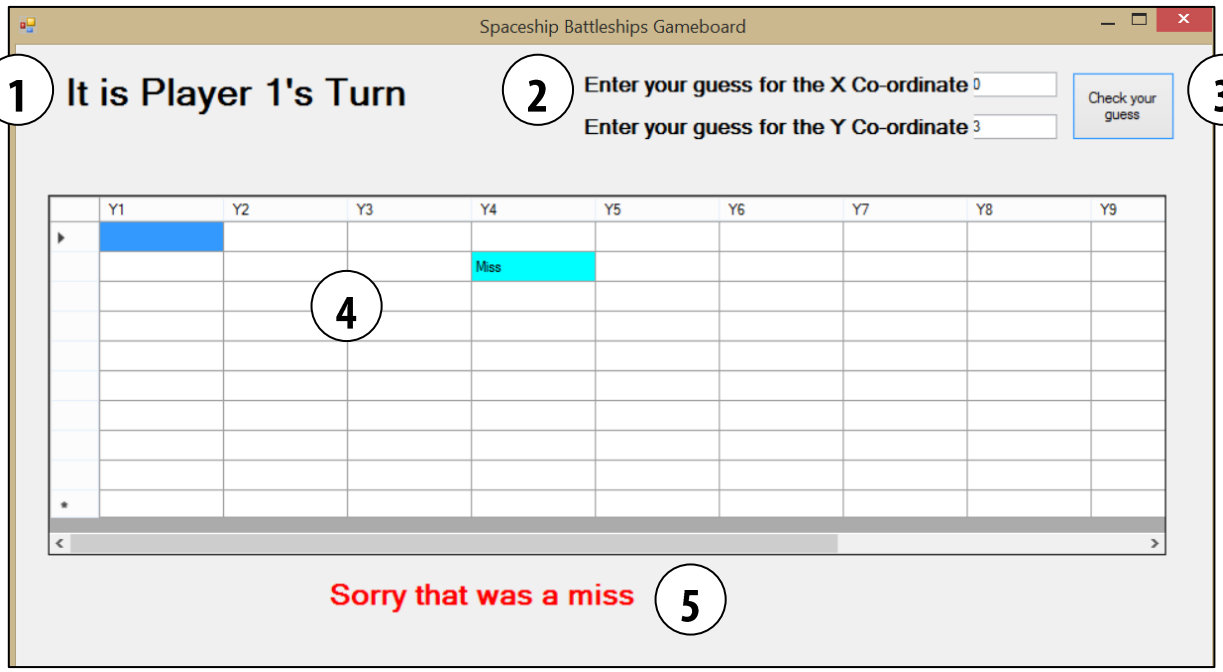
```
CLASS Bomber Inherits Spaceship
Public Function GetSpaceshipType Overrides
Public Sub SetNumberofPositions
-----------------------------------
Private Type : String
Private NumberofPositions: integer
```


4.5 – Interface design

In this section you need to show a **selection of the main screens** you intend to use for an interface. This should include screens that accept input as well as screens that show output. Although at one time sketches of how the interface would look was required now it is perfectly acceptable to actually build the interface in the programming environment you intend to use. In this way they are prototype screens. The purpose is to show how your end user will interact with the program through the interface and to explain your interface design reasoning in relation to the needs of your end user.

The example **Fig 19** shows one prototype screen for the Spaceships game and how the design of it might be explained

Figure 19 – Spaceships: An example of a Prototype screen design and interface explanation



1. When the form loads (using the form load event) this label will display whether it is player 1 or player 2's turn. The algorithm placed on the load event will randomly choose either player 1 or player two to start first.
2. The player will insert the co-ordinates into the text boxes where they want to guess at the position of the other player's spaceships. There will be validation placed on this to ensure that the co-ordinates that are entered are valid for the grid.
3. When the player clicks the button it will pass the entered co-ordinates as arguments to the Rows.Cells. methods of the grid.
4. This will result in the backcolour of the cell changing (Aqua for a miss, red for a hit, green if a whole ship has now been destroyed) and either the text "Miss", "hit" or "destroyed" appearing in the cell.
5. Finally, a message will be output to the label below the grid to make it clear whether the guess was a hit, a sinking or a miss. A timer will hold the message on the screen for 10 seconds before the player1 grid goes invisible and the player 2 grid becomes visible. At the same time the label showing the players name will change to the other player and the guess co-ordinate textboxes will clear.

It is also a good idea to briefly justify your choice of a particular type of interface. Some examples you might decide to use are:

- Form or windows interface (Graphical User Interface)
- Command line interface (Console window interface)
- Menu-driven interface
- Web interface

4.6 – Algorithms

This is arguably the most important section in your design and certainly you cannot achieve high marks in this section without it being completed to a high standard. Your algorithm design should cover the main processing objectives for your project and you should endeavour to use standard pseudocode. They should not be written in any particular programming language.

As a guide when describing your algorithms, you should:

- State what part the algorithm will play in your program (what objective will it satisfy)
- Describe the algorithm in pseudo-code

For complex algorithms you should explain how the different parts of the algorithm work.

Example 3 shows part of an algorithm that will check for a hit in the Spaceships program.

Example 3 – Spaceships: Example of an algorithm described in pseudo-code

```
SUB PROCEDURE HitCheck (HitBoard (,) ShipBoard (,) PlayerNumber)
 IF ShipBoard(x, y) NOT " " THEN

 IF ShipBoard(x, y) ⟵ "F" THEN
           PlayerStats(PlayerNumber).FighterHealth ⟵ destroyed
           PlayernumberGrid.Rows(x).Cells(y).Backcolor ⟵ RED
           PlayernumberGrid.Rows(x).Cells(y).Value ⟵ "destroyed"
  Shipboard(x, y) ⟵ D
  OUTPUT MESSAGE "Well done the spaceship is destroyed

  ELSEIF ShipBoard(x, y) ⟵ "B" THEN
   PlayerStats(Playernumber).Positions ⟵ Positions - 1
        CALL CheckBomberSunk(PlayerNumber, PlayerStats, Shipboard)
        IF Bombersunk ⟵ true THEN
             PlayernumberGrid.Rows(x).Cells(y).Backcolor ⟵ GREEN
        PlayernumberGrid.Rows(x).Cells(y).Value ⟵ "destroyed"
   Shipboard(x, y) ⟵ D
        OUPUT MESSAGE "Well done the spaceship is destroyed"
        ELSE
             PlayernumberGrid.Rows(x).Cells(y).Backcolor ⟵ RED
             PlayernumberGrid.Rows(x).Cells(y).Value ⟵ "hit"
    Shipboard(x, y) ⟵ H
   OUTPUT MESSAGE "Well done - that is a hit"

END SUB
```


Documented Design: Evidence Checklist

Key to the importance of sections of evidence:

- Any item marked with an **E is essential** and **must be included** if you are to gain reasonable marks in this section
- Any item marked with a **D is desirable** and **should be included,** it will help to gain good marks but is not always essential
- Any item marked with a **C is complementary**. It may help explain aspects of your project but is not essential

Item No	Have you included???	Tick
1	**For Data-Handling projects only:** • ERD and EAM **[E]** • Normalisation **[E]** • Data Dictionaries **[E]** • SQL statements **[E]** • Data Flow Diagrams (to show main data processing) **[D]**	☐
2	**Overall System Design** • Top Down diagram **[E]** • IPSO chart **[E]** • Flowcharts to show main processing (non-data handling projects) **[D]**	☐
3	**Data dictionaries / Record Structure / Data Structures [E]**	☐
4	**OOP Class Design (if OOP is used) [E]**	☐
5	**Interface Design [E]**	☐
6	**Algorithms [E]**	☐
7	**Hardware Selection / Design (if appropriate) [D]**	☐
8	**Security and Integrity of Data [C]**	☐

Documented Design: Assessment Criteria

There is a maximum of **12 marks** awarded for this section

Mark band	Advice	Mark range
Upper Mark band	A third party would be able to understand how **all the key parts** of the system are supposed to work – overview, interface, data dictionary, data structures, class design (if OOP). Algorithms are present that cover **all** the key processing areas. **For a data-handling project** there is accurate normalisation and description of tables that produces a workable data structure for the problem. There is a **range of SQL statements** that covers all the **key data-handling requirements**.	10–12
Upper Middle Mark band	A third party would be able to understand how most of the key elements of the system are supposed to work - overview, interface, data dictionary, data structures, class design (if OOP). Algorithms are present that cover **most of** the key processing areas. **For a data-handling project** there is a good attempt to normalise relations, describe the structure to be built and SQL to be used.	7–9
Lower Middle Mark band	A third party would be able to understand how **some key parts** of the system are supposed to work (half or less). Some algorithms are included. For a data-handling project there is some attempt to normalise relations, describe the structure to be built and SQL to be used.	4–6
Lower Mark band	This would be a **difficult design** for a third party to understand as **key information** (algorithms, data dictionaries, data structures) is **missing** or incomplete	0–3

5. Technical Solution

The purpose of the section of your project report is to show that you have built a solution to meet your objectives (set out in Analysis). This is the section with the largest marks so you should ensure that this is thoroughly documented. If you are producing your technical solution as a series of prototypes make sure you read the chapter on prototyping first.

5.1 – Documentation advice specific to data-handling projects

If your project involves any data handling using a relational data structure (two or more related tables) then you should provide evidence of:

Evidence of setting up tables and setting relationships

Because there is a requirement for a fully coded solution – the setting up of the tables and relationships should be done using DDL statements embedded in whichever programming language you are using to create your solution. (CREATE TABLE). See **Example 4** (created in VB.NET):

Example 4 – An example of code used to create a relational database structure

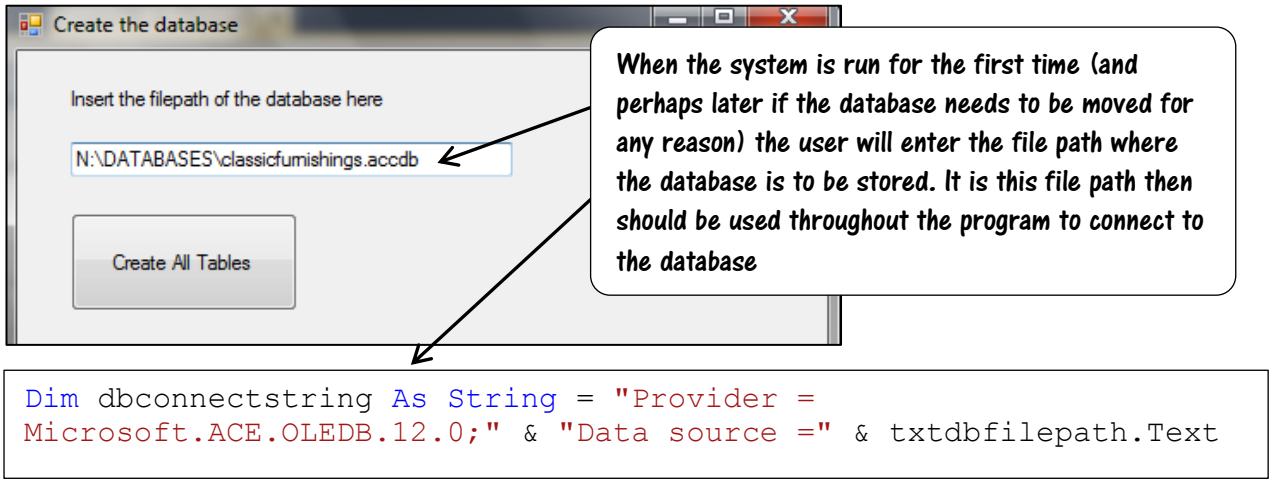
```
Private Sub createorderstable(ByRef dbconnectstring As String)
        'the connection provider and filepath to the database (data source) are passed
        as a parameter from the procedure where the button on the form is clicked to
        create the tables
        Using con As New OleDbConnection(dbconnectstring)
            con.Open()
            'the connection to the database is opened
            Using cmd As New OleDbCommand
                cmd.Connection = con
                'the command that will create the tables is set to use the provider and
                filepath passed as a parameter to connect with
                cmd.CommandText = "CREATE TABLE TBLOrders(OrderID INT NOT NULL PRIMARY
                KEY,OrderMadeDate DATE NOT NULL,OrderTotal CURRENCY,OrderVAT
                CURRENCY,LabourCost CURRENCY,DateCompleted DATE, CustomerID INT
                CONSTRAINT FK_CustomerID REFERENCES tblCustomers (CustomerID))"
                'this is the command that will be executed
                Try
                    cmd.ExecuteNonQuery()
                    'the create table statement is executed within the database
                    lblcreatedmessage.Text = "Table created"
                    'a message is output to the label if there are no exceptions
                Catch ex As Exception
                'if an error/ exception occurs then an error message is displayed
                    lblcreatedmessage.Text = ex.Message
                End Try
            End Using
            con.Close()
            'the connection to the database is closed
        End Using
```

Obviously there will be a need for this script to run once during the setting up and installation of the program although depending on the objectives for your project it could be done again at some other point. However

you do need to show in your solution (or in testing) that this element has produced the tables because you will need to input and manipulate data into and from those tables.

Additionally it is a good idea to enable the file path of the database to be set/ manipulated dynamically from the interface of your system (**see Fig 20 below**). If you fix the file path in the code to your own file structure/system this prevents the ultimate user from placing the database component within their own file structure.

Figure 20 – Classic Furnishings: Setting the Filepath



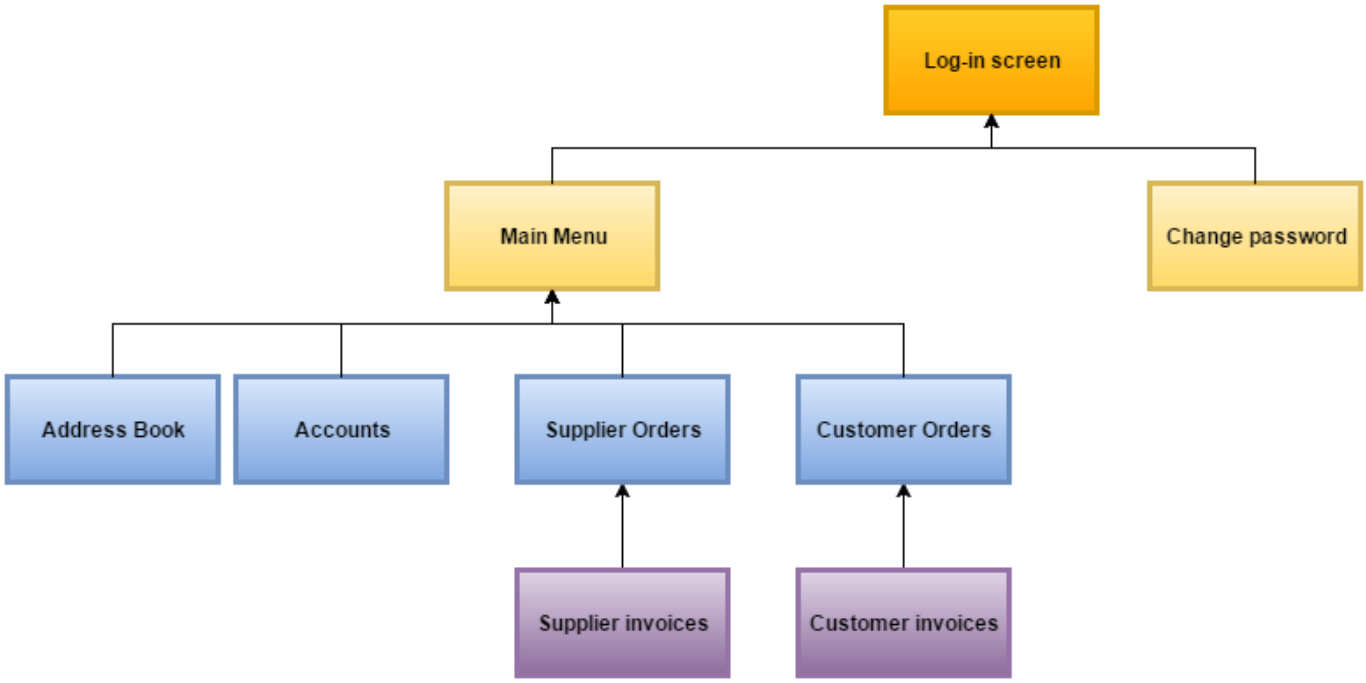

```
Dim dbconnectstring As String = "Provider =
Microsoft.ACE.OLEDB.12.0;" & "Data source =" & txtdbfilepath.Text
```

5.2 – System Overview

Show your system overview diagram or top-down diagram of your built system. This gives the marker /moderator a good overview of how the different parts of your built system link together.

Fig 21 shows the system overview for the Classic Furnishings example used previously in the booklet

Figure 21 – Classic Furnishings: System Overview

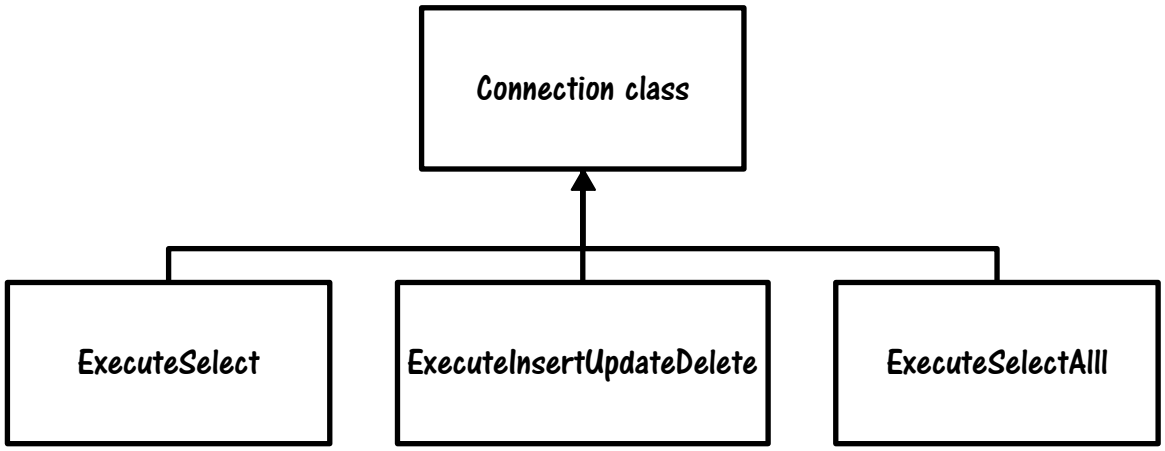


5.3 – Evidence of built classes

If you have built classes which are subsequently used in the rest of your code then completely list the code for them. You should make sure that your classes are adequately annotated to describe how your methods work. Additionally it is a good idea to show the class diagram (**Fig 22**) before the related code listing.

Figure 22 – Example of how to display evidence of classes used in the solution.

Of course this example just shows one of the classes but you would need to show all your built classes (i.e. classes you have coded yourself)



This class acts as the parent class that is inherited by the other database classes. This class contains the details of the connection to the database and the method to set the SQL statement to execute.

```
Imports System.Data.OleDb
Public Class Connection
    Protected connection As New
    OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;" & databasefilepath)
    'Declares a variable that can only be accessed in this class or by inherited
    classes - this variable stores text enabling the connection to the
    database(connecting string)
    Protected dbcommand As New OleDbCommand
    'Declares a variable that can only be accessed in this class or by inherited
    classes.This variable stores the sql command to be executed

    Public WriteOnly Property retrievesql() As String
        Set(ByVal sql As String)
            'the parameter that is passed to this property is to the variable sql
            connection.Close()
            'closes a connection if there is a open connection
            dbcommand.CommandText = sql
            'sets the sql command to be executed
            dbcommand.Connection = connection
            'sets the connecting string to be used to connect to the database
        End Set
    End Property
End Class
```

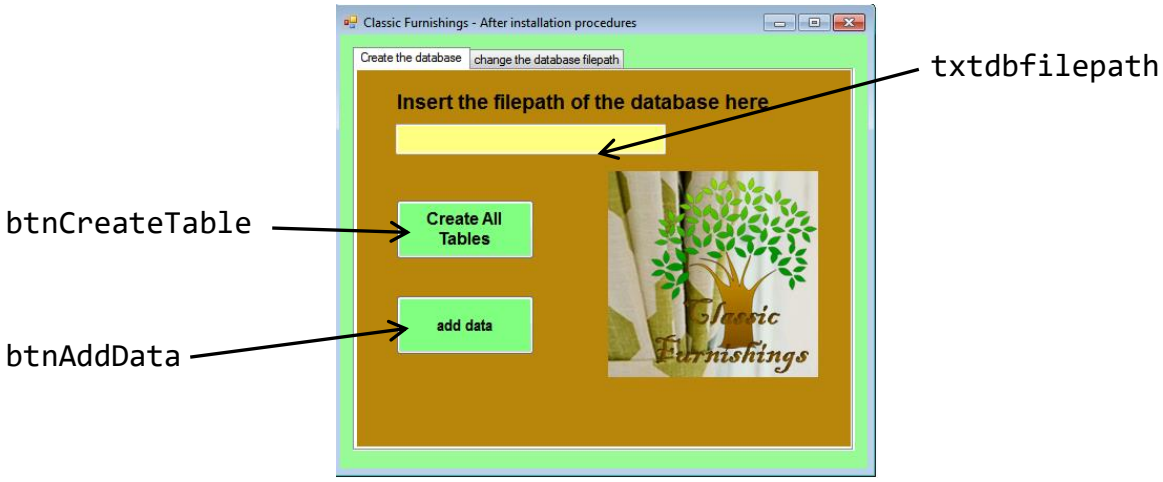

5.4 – Evidence of complete code listings and user Interface

The purpose of this section of evidence is to show your complete code listing for all parts of your project. **If your project modules are built around forms or windows** so that each form has a module of code attached to it then it is a good idea to show a screenshot of the form followed by the code listing for the module attached to that form (see example 5).

Annotating your code – the purpose of annotation is to explain how your code works to someone who has not seen you developing your program. Whilst it is not necessary to annotate every single line of code you should ensure that code is adequately annotated to explain how small sections work. This is particularly true of complex sections of code which will need to be more thoroughly annotated than the more straightforward sections. In the Function Retrievedirectory in example 5 each loop has been annotated to explain how the function extracts the directory path from the whole filepath.

Example 5 – an example of documentation of a form and its associated code module

(NB. only some procedures shown here whereas you should list all your code)




```
Imports System.Data.OleDb
Public Class Dbcreate
 Dim textfiledir As String

 Private Sub btnCreateTables_Click_1(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCreateTables.Click
 Dim dbconnectstring As String = "Provider =
Microsoft.ACE.OLEDB.12.0;" & "Data source =" &
txtdbfilepath.Text
 Dim filepath As String = txtdbfilepath.Text
 Call retrievedirectory(filepath) ' this retireves a
directory path for the textfile
 Call SaveFilepath(filepath, textfiledir) ' this saves
the filepath of the database as a text string (in a
text file) in the same directory as the database
 Call CreateCustomerTable(dbconnectstring) ' this
creates the customer table
 lblmessage.Text = "" ' clears the message label
 Call createorderstable(dbconnectstring) 'creates the
orders table and associated relations
 End Sub

 Private Function retrievedirectory(ByRef filepath As
String)
 Dim directory(filepath.Length) As Char
 Dim count As Integer
 Dim count2 As Integer
 Dim dirpath As String = ""
 For i = 0 To filepath.Length - 1
  directory(i) = filepath.Substring(i, 1)
  'assigns all the characters in the filepath string to
an array (that has the same number of indicies as there
are characters in the filepath)
 Next
 count = directory.Length - 1

 Do Until count = 0 Or directory(count) = "\"
  directory(count) = "Z"
  count = count - 1
  'this replaces all the characters after the last
backslash to Z's
 Loop
 count2 = 0
 Do Until directory(count2) = "Z"
  dirpath = dirpath & directory(count2)
  count2 = count2 + 1
  'this concatenates a string made of all the
characters in the array that make up the directory path
(not including the characters that make up the filename
and extension)

 Loop
 textfiledir = dirpath
 Return textfiledir
 'the directory path for the database is returned to
the calling procedure
 End Function
 End Class
```


5.5 – Evidence of procedures and variables

This should be a complete listing of the procedures and variables used in your code. This is best presented as table with columns like **example 6**.

Example 6 – How procedures and variables might be described for example 5

Form DBCreate – This form will be accessible from the main menu. It will be used after installation to set the filepath for the database and create the required tables		
Global Variables	**Textfiledir (*holds the location of the text file that will hold the filepath of the database for use when the connection string is needed*)**	
Procedures	**Purpose**	**Variables**
Sub CreateTables_Click	**When the Create tables button is clicked this procedure will create the required tables in the database**	**dbconnectstring** **filepath**
Sub CreateCustomerTable	**Creates the customer table in the database when called by CreateTables**	**Con** **cmd**
Sub CreateOrdersTable	**Creates the Orders table in the database when called by CreateTables**	**Con** **cmd**
Sub SaveFilepath	**This procedure writes the text file to the location specified by the function RetrieveDirectory**	**Filepath (parameter passed)** **Textfiledir (parameter passed)** **Textfilepath** **objwriter**
Function Retrievedirectory	**This function calculates from the entered filepath of the database, the directory where the text file is to be stored (same directory as the database). It returns the directory path to CreateTables and is used in SaveFilepath.**	**Filepath (parameter passed)** **Directory(array)** **Count** **Count2** **dirpath**

How your project will be marked

The AQA specification lists the required skills for each mark band and you should consult this alongside this guide to ensure that your project has the desired technical competence.

Technical Solution: Evidence Checklist

Key to the importance of sections of evidence:

- Any item marked with an **E is essential** and **must be included** if you are to gain good marks in this section
- Any item marked with a **D is desirable** and **should be included,** it will help to gain good marks but is not always essential
- Any item marked with a **C is complementary**. It may help explain aspects of your project but is not essential

Items 6 and 7 in the list are not really part of your evidence but are things you should take into account as they have an impact on how your project is marked (See "Applying the Assessment Criteria" below).

Item No	Have you included???	Tick
1	**For Data-Handling projects only:** • Evidence of CREATE TABLE statements (for first run of the program) • Evidence of created tables and accurately set relationships **[E]**	☐
2	**OOP Built classes (if you have them) [E]**	☐
3	**Interface Screenshots [D]**	☐
4	**Complete Annotated Code listings [E]**	☐
5	**Procedures and variables Listings [D]**	☐
6	**How complete is my solution?** Check how many of your original objectives have you achieved *** You are awarded marks out of 15 based on how complete your solution is. This is not just about how many of your original objectives you have completed but is also about what objectives you could reasonably be expected to have for a problem or investigation of the type identified in your Analysis.* *For example, a student has completed a solution for a Quiz program to help students revise for a subject. The student achieves all the original objectives set in Analysis but failed to include an objective that would allow the teacher to add new questions to the program. This solution could not then be awarded full marks for the solution (even though all the original objectives had been achieved) as it is reasonable to expect a revision system to include a facility for updating/ replacing/ adding new questions – otherwise such a system becomes redundant pretty quickly*	☐
7	**Skills check (**see table below and tick off the skills you think you have achieved OR ask your teacher/tutor/NEA supervisor to tick the skills they think you have achieved**)**	☐

Skills Checklist

Top Mark band (excellent programmer)	**Middle mark band** (good programmer)	**Lower Mark band** (some programming skill)
☐ Candidate written classes (OOP) ☐ Complex standard algorithms and structures (structures, arrays of records, stacks, queues, linked lists, trees and graphs) ☐ Use of regular expressions ☐ Use of hashing techniques ☐ Use of recursion ☐ Merge sorting or equivalent	☐ Use of arrays (2-D) ☐ Some standard algorithms like bubble sorts and binary searches ☐ Simple OOP classes ☐ Simple client-server model **For Data- handling solutions** ☐ Simple data model with a few inter-linked tables	☐ 1-D arrays ☐ Uses a small range of different data types ☐ Uses linear searching methods ☐ Uses simple mathematical calculations ☐ Accesses table data without the use of SQL

☐ Implementing complex mathematical processes (KS5 Maths at least) ☐ Time simulation / scheduling ☐ Complex client server model **Specific to Data- handling solutions** ☐ A complex data model with many related tables (multiple normalised entities to at least 3rd normal form) ☐ Using parameterised cross-table SQL statements embedded in the programming language ☐ User generated Data Definition statements (e.g. CREATE TABLE embedded in the code)	☐ Single-table SQL statements ☐ Use of text files for storing and retrieving data	

Coding Style Checklist		
Top Mark band (excellent coding style)	**Middle mark band** (good coding style)	**Lower Mark band** (basic coding style)
☐ Code is split into appropriate subroutines (procedures and functions with parameters passed where appropriate) ☐ Subroutines have a single purpose ☐ Subroutines whose purpose are related are grouped together ☐ Good use of defensive programming – attempting to deal with issues that might cause your program to crash. ☐ There is good use of exception handling	☐ Use of subroutines ☐ Well-designed user interface ☐ Minimal use of global variables ☐ Appropriate use of local variables ☐ Any casting (data type conversion) needed is managed by the programmer ☐ Constants are used where appropriate ☐ File paths are parameterised (not hard coded into the solution) ☐ Code is appropriately indented ☐ Code is appropriately annotated	☐ Meaningful identifier names are used for variables / structures / procedures, from controls etc.) ☐ Code is annotated to explain how sections of code work.

Technical Solution: Assessment Criteria

There is a maximum of **42 marks** awarded for this section. This is based on two strands of evidence (Skill and coding style AND completeness of solution)

1. How **skilful** is my solution (check which skill band your solution mostly falls into from the **skills table** above) and then:
   a. What is the **quality of my coding style**? This will help to determine where you are placed in the **SKILLS** mark band. For example if you have achieved a good range of the skills in the "excellent programmer" band in the skills table and mostly "good coding style" in the coding style table then you will be placed towards the middle of the range 19-27. The coding style is cumulative. To be considered to have excellent coding style you must also have satisfied the appropriate criteria in the lower two coding style bands.

2. How **complete** is my solution? The remaining 15 marks are awarded for the completeness of your solution – how many of your original objectives you have achieved. Remember that completeness is not just about the objectives you have set but what should have been reasonably set for a project of this type.

Skill and Coding Style

Mark band	Skill level achieved	Mark range
Upper Mark band	1. The **programming skills used are mostly Top mark band** (very good programmer) with a. **Excellent coding style** (towards the top of the mark range 25–27). This means that all the good coding techniques and all / nearly all of the excellent coding techniques have been used. b. **Good coding style** (middle of the mark range 22–24). This means that all the good coding techniques have been used and a majority of the excellent techniques. c. **Adequate coding style** (lower end of the mark range 19–21). This means that coding techniques used are mostly good with some excellent characteristics.	19–27
Middle Mark band	2. **The programming skills used are mostly middle mark band** (fair programmer) a. **Excellent coding style** (towards the top of the mark range, 16–18). This means that all the good coding techniques have been used and a majority of the excellent techniques b. **Good coding style** (middle of the mark range, 13–15). This means that coding techniques used are mostly good with some excellent characteristics. c. **Adequate coding style** (lower end of the mark range, 10–12). All to almost all of the good coding techniques with perhaps one example of **an** excellent technique	10–18
Lower Mark band	3. The programming techniques used are mostly lower mark band (some programming skill) a. **Excellent coding style** (towards the top of the mark range, 7–9). All to almost all of the **good** coding techniques. b. **Good coding style** (middle of the mark range, 4–6). Uses some good coding style **characteristics**. c. **Adequate coding style** (lower end of the mark range, 0–3), uses basic coding techniques.	0–9

Completeness of Solution

Mark band	Completeness of Solution Criteria	Mark range
Upper Mark band	A solution that **meets all / nearly all of the objectives** of the system *(including those that might reasonably be expected for a system of this type which you may or may not have included in your original objectives)*	11–15
Middle Mark band	A solution that meet **many of the objectives of the system** (including some of the important ones) *(including those that might reasonably be expected for a system of this type which you may or may not have included in your original objectives)*	6–10
Lower Mark band	A solution that has made an attempt to **tackle some parts** of the system *(including those objectives that might reasonably be expected for a system of this type which you may or may not have included in your original objectives)*	0–5

6. Testing

6.1 – What needs testing?

This is the section of your project where you prove that your solution works as intended and therefore that you have met your objectives set out in analysis. So you need to draw up a test plan of what needs to be tested (and how it will be tested) carefully in order to show clearly that objectives have been achieved. It needs to cover your key processing objectives including:

- That any planned validation of input works as intended
- That valid inputs are accepted by the system
- That data is processed to produce the intended outputs
- That any planned navigation works as intended
- That the system works as a whole system

6.2 – Choosing a testing strategy

It is a good idea to set out (briefly) your testing strategy. What methods of testing do you intend to use to prove your solution works? If you are unsure of what methods are available consult your A Level textbook. Some common methods used for testing (in A-Level projects) are listed below in **Table 12** although this list is not exhaustive.

Table 12 – Common testing methods

Testing Method	Description of how it is applied
Black-box testing	Concerned with testing interaction with the interface. Do the inputs produce the required outputs?
White box testing	This is also known as structural testing to test every possible logical path through individual modules of code. Might include the use of trace tables
Module testing	This is testing individual components of a system (like procedures or functions)
End-User testing	Often know as beta testing. The end users use your system with real data and report any faults or errors

6.3 – Setting out a test plan

It is considered good practice to use a testing table to plan out your tests. **Table 13** shows the suggested headings you should use as well as an example of an entered test. You may wish to add an extra column for additional notes on any failed tests or this can be done under a separate heading in your testing document.

Table 13 – An example test plan showing suggested headings and a test example

Test No	1
Objective / Requirement No	1a
Purpose of test	To ensure that players of the spaceships game can only enter valid co-ordinates (between 1 and 10) when choosing a grid location on the game board to fire a missile at
Description of test	When the game asks for co-ordinates to be entered – the stated test data will be input
Test data	A) T = 5,8 B) Er = 4,11 C) Ex = (1,10), (0,11), (2,9)
Expected result	Typical data should be accepted and a missile fired at that location Erroneous should be rejected and an invalid input message output. User should be prompted to re-enter valid co-ordinates Extreme data – data within and on the boundaries (1, 10 and 2, 9) should be accepted and a missile fired at that location. Data outside the boundary (0,11) should be rejected the same as erroneous data
Actual result	Typical data = pass Erroneous data = pass Extreme data (1, 10) and (2, 9) = pass (0,11) = pass

Key to Test data types: T = typical, Er = Erroneous (abnormal), Ex = extreme (boundary)

6.4 – The different types of test data

There are three types of data that can be used in testing:

1. **Typical** (normal) data – valid data that would normally be entered for a specific feature
2. **Erroneous** (abnormal) – invalid data for a specific feature
3. **Extreme** (boundary) data – data that is just before, on and just after the outer limits of a valid range. ,e.g. if a valid range is 1-5 then extreme test data is 0, 1, 2, 4, 5, and 6. If a valid input has to be >=10 then extreme test data would be 9, 10 and 11.

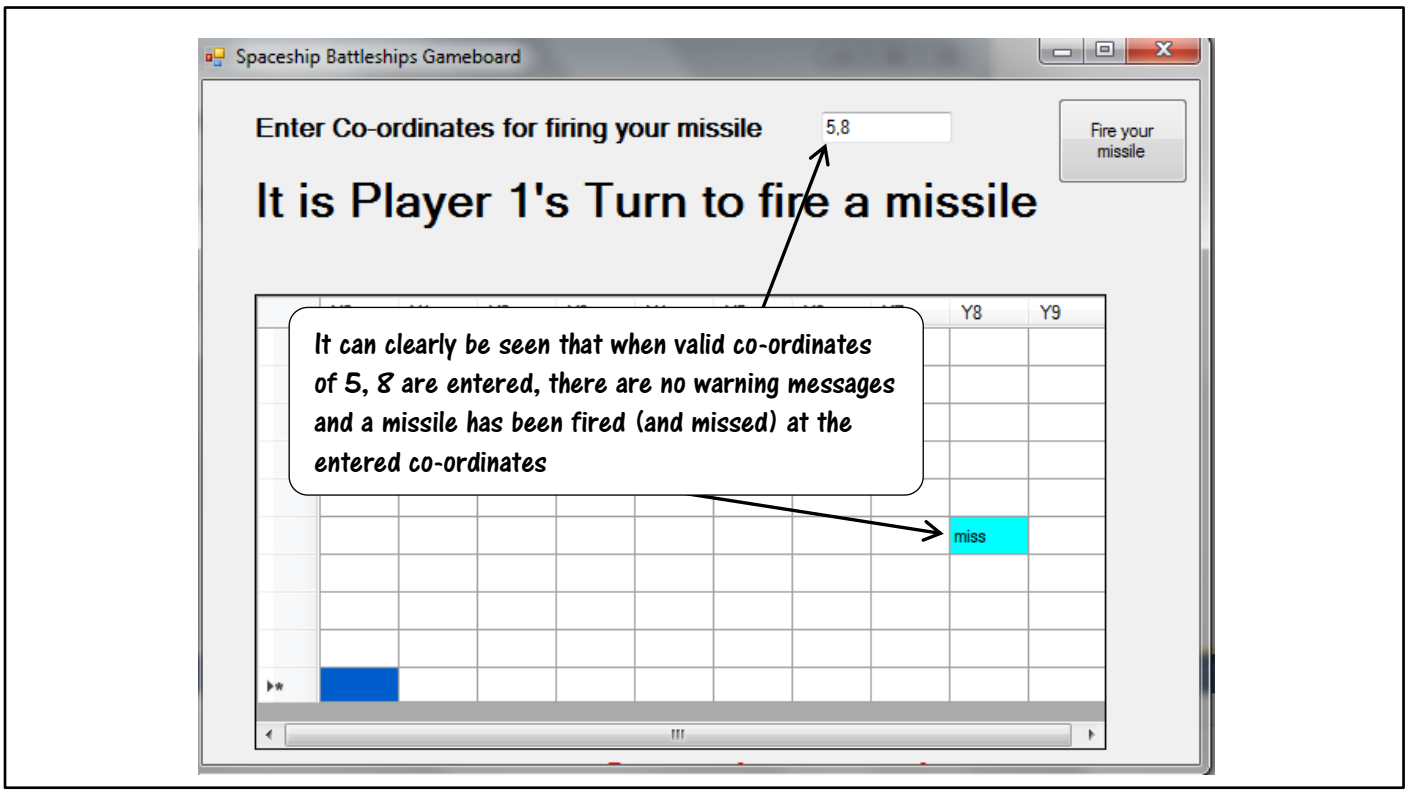
It is generally a good idea to use all three types of test data – typical, erroneous (abnormal) and extreme (boundary). You should try to use extreme (boundary) data **where possible** in your testing.

6.5 – Test evidence

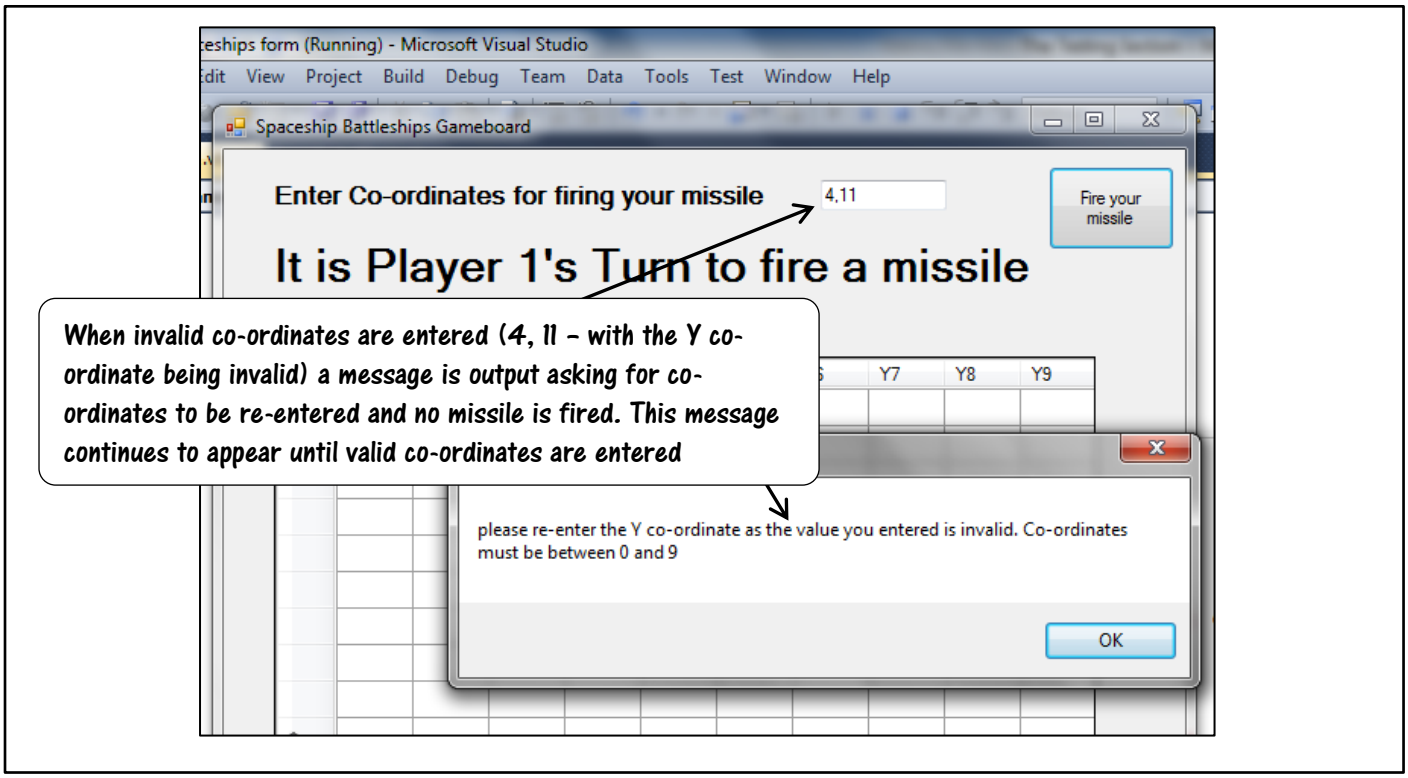
You need to provide annotated and cross-referenced screenshots of the actual results produced by your tests. These screenshots should clearly show the input that led to the processed output. See **example 5 and 6**.

It is possible to refer to testing screenshots provided in your technical solution as long as these screenshots are cross-referenced with a test number next to the screenshot and the page number of the screenshot is placed in the test plan.

Example 5 – Test number 1a (typical data) from Table 13



Example 6 – Test evidence for test 1b (erroneous data) from Table 13



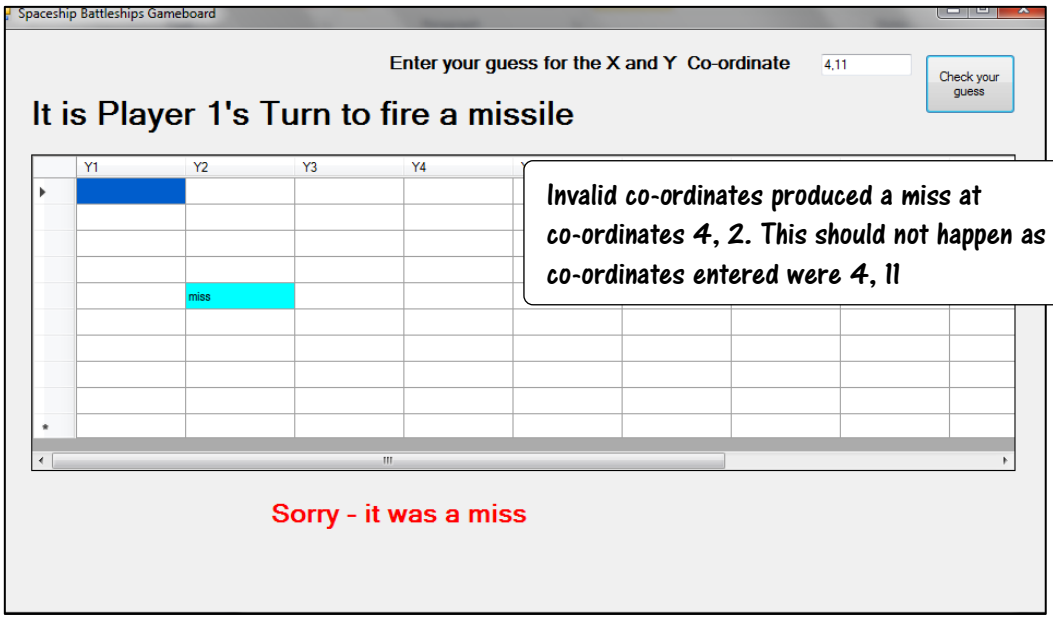
Video evidence

There are some systems that do not readily lend themselves to taking screenshots that prove the system works (e.g. control systems for robots, games). **Consider taking some video evidence or photographs** that prove the required outcomes were achieved. Video evidence is also useful for proving that the system works as a whole. Video evidence should be uploaded to YouTube and a link provided in your documentation. Any video evidence of testing does need to be explained though. This can be done by overlaying the video with an explanatory soundtrack or text explanations or by providing a written explanation to go with each video test in your documentation.

Failed tests

If you have any tests that do not produce the expected results and are therefore classified as a fail – try and fix the issue that is stopping the test from passing. Document any changes you make to the code, and possibly the interface, in order to make this happen. If you do manage to fix the issue, then re-run the test, showing that the desired result has been achieved. **Example 7** below how a failed (and then fixed) test might be documented.

Example 7 – An example of how to document (with screenshots) a test that fails to work



This seems to be due to the method I have used to find the individual co-ordinates from the string entered. For the Y co-ordinate it is only taking the first character of the string (1) and I have the Y co-ordinates numbered 1-10 rather than 0-9.

```
guess = txtguess.Text
X = CInt(guess.Substring(0, 1))
MsgBox(X)
Y = CInt(guess.Substring(2, 1))
```

I have now fixed this problem by using a split string function instead – this splits the entered string at the "," so no numbers are lost off the X and Y co-ordinates

```
guess = txtguess.Text
coordinates = guess.Split(",")

X = CInt(coordinates(0))
MsgBox(X)
Y = CInt(coordinates(1))
```


6.6 – Testing qualitative objectives

If you have any objectives that are qualitative then these may need to be tested by your end user. ,e.g. An objective that states that the interface must be user –friendly. This can only really be judged by the end user as what is user friendly to one person might be incomprehensible to someone else. Provide a separate test plan for this and get feedback from your user on the outcomes of the tests carried out. See **Table 14.**

Table 14 – Example of end user testing from the Spaceships scenario

Test No	Objective / Requirement No	Purpose of test	Description of test	Test data	Expected result	Actual result / End user comments
1	8	To test whether the end user finds the help message pop-ups useful when trying to enter valid co-ordinates	When the end user goes to enter the co-ordinates they want to fire a missile at , the following messages appear: When the co-ordinate textbox gains focus a message appears prompting entry within the valid range.	Typical (user chooses valid co-ordinates)	Pop-up message appears on textbox gaining focus ("You must enter co-ordinates between 0 and 9"). User finds this prompt useful and then can correctly enter valid co-ordinates	I entered co-ordinates (5 8) and this was rejected as invalid. *Additional information is needed on the help message about the format expected or additional code that will process the co-ordinates without a comma in-between*

If your end user does make reasonable suggestions at this point then it is a good idea to try and implement those changes, making sure that you document your changes and include them in your testing section. Re-run the test with your end user and document their fresh response.

Testing: Evidence Checklist

Key to the importance of sections of evidence:

- Any item marked with an **E is essential** and **must be included** if you are to gain good marks in this section
- Any item marked with a **D is desirable** and **should be included,** it will help to gain good marks but is not always essential
- Any item marked with a **C is complementary**. It may help explain aspects of your project but is not essential

Item No	Have you included???	Tick
1	**My testing strategy [C]**	☐
2	**My test plan:** • **Including a description of the test** • **What objective it covers** • **What test data I will use and what type of data it is** • **The expected result** • **The actual result [E]** Make sure you include typical, erroneous and extreme data (where appropriate).	☐
3	**Testing screenshots (or video evidence) cross referenced to the test plan and annotated to describe what happened when the test was run. [E]**	☐
4	**Any improvements that were made to the solution as a result of failed tests [E]**	☐
5	**End-user testing (particularly of qualitative objectives) [D – useful in Evaluation]**	☐

Testing: Assessment Criteria

Mark band	Advice	Mark range
Upper Mark band	The solution is **thoroughly tested** proving that the complete (or nearly complete) solution is robust (doesn't crash). To achieve a mark in this range your testing must prove that the **system as a whole** works as intended. Test evidence must be clearly and thoroughly **explained** It proves that the objectives identified in analysis have been achieved	7–8
Upper Middle Mark band	A **good, fairly extensive, range of testing** has been carried out but it is not obvious that all the core objectives have been achieved. This might be because some key tests are not clear enough (poor screenshots, limited annotation of screenshots or poorly defined tests) or because some key elements have not been tested The testing evidence must be **explained**	5–6
Lower Middle Mark band	The system has been **mostly tested** but does not demonstrate that the key objectives have been achieved and the solution is robust There is an **attempt to explain** the evidence	3–4
Lower Mark band	**Some parts of the system** have been tested which demonstrates that some of the objectives have been achieved	1–2

7. Evaluation

*In this section of work you are **evaluating the success** of the objectives you set out in your analysis section. If you set performance criteria for your objectives then you use these to help judge the success or otherwise of your solution.*

7.1 – Comparing performance against the objectives

It is possible to simply copy over from analysis the table where you set the original objectives and performance criteria. Add an extra column to the table so that you can evaluate how successful your actual solution was in meeting each individual objective. It is useful to state whether or not this objective was completely achieved. Alternatively each numbered objective should be stated with an explanation in a following paragraph of what was achieved and how well it was achieved. **Table 15** shows how **one** particular objective has been evaluated. Here the objective was completely successful so the emphasis in the evaluation is about specifying exactly *how well* the performance criteria were met.

Table 15 – Original objective number 1 from the Classic Furnishings Scenario
(see Analysis chapter – Table 6)

No	Objective	Performance Criteria	Evaluation
1	Create a secure storage area for customer details	This must be secured so that only the end user can access this data.	- COMPLETELY ACHIEVED - The system I built is password protected so that only the client can access the system. Initially she will need to log in with the pre-set password but will be prompted to change it to a strong password on first use. The system will insist that passwords must be more than 8 characters long using lowercase letters, at least one upper case letter, numbers and some special symbols out of a pre-set list. Additionally the system allows the user to password protect the database with a password independently of the system in case the database file is compromised rather than the system. The system allows the client to change the password on the database if she feels that the set password might be compromised. As a further level of security all sensitive data (password information, client data) stored in the database has been encrypted. Testing shows that it is not possible to log in to the system *(page reference)* or open the database *(page reference)* without the correct password. Nor is it possible to set a weak password *(page reference)*. Testing screenshots of the database also prove that password data and client data has been encrypted and made unreadable *(page reference)*.

Objectives that are not met or partially met

With objectives that are partially met, the emphasis will be on explaining what works and what doesn't. **Table 16** shows the objective from **table 15** if it had been partially met. The changes from **table 15** are shown in italics. If you have any objectives that have not been met you should try and explain what you tried to do to achieve this objective and what hurdles prevented you from completing it in the end.

IMPORTANT TIP!
There is often a temptation, when some objectives have not been achieved to actually remove them from the project altogether. However you must be careful as this may adversely affect marks in Analysis (and possibly the technical solution –completeness strand) as your objectives will no longer cover all the needs of your end user (as specified in the assessment criteria).It is recommended that you seek your teacher's advice about the advisability of removing objectives and what impact it may have on marks in other areas of your project.

Table 16 – Example of an objective that has been partially achieved

No	Objective	Performance Criteria	Evaluation
1	Create a secure storage area for customer details	This must be secured so that only the end user can access this data.	- PARTIALLY ACHIEVED - The system I built is password protected so that only the client can access the system. Initially she will need to log in with the pre-set password but will be prompted to change it to a strong password on first use. The system will insist that passwords must be more than 8 characters long using lowercase letters, at least one upper case letter, numbers and some special symbols out of a pre-set list. Testing shows that it is not possible to log in to the system without the correct password. Nor is it possible to set a weak password. *However although the system is secure the database file itself is not. This could be opened independently from the system and the data inside could be accessed and even edited using database software. I did try putting a password on the database but then had problems with the database connection string so I had to remove the password from the database before the connection string would work. (Note: this would probably be because the password was not included in the connection string)*

7.2 – Getting and analysing independent feedback

As well as completing your own evaluation of your objectives you also need to obtain feedback from your end user on how well they think your system met their identified needs. This can be done through a combination of end-user testing and end-user evaluation. Some qualitative objectives can really only be evaluated by your end user. For example any objective that states the interface must be user-friendly can only be tested for user friendliness and evaluated as to success by the end user or someone who will be using the system built.

Table 17 shows an extract from End-User evaluation of the classic furnishings scenario

No	Objective	Performance Criteria	Evaluation
1	The end user must be able to create the database when the system is run for the first time after installation	The end user can enter the location of the database file and create the database tables on first run of the program after installation	"I found this feature hard to use initially as it was not creating the database tables. I realised after some trial and error that it was because I wasn't using the correct syntax in the filepath I was inputting. How the filepath should be constructed needs to be made more obvious – especially if we ever have cause to recreate the tables again

7.3 – How could the outcomes be improved?

Once your end user has evaluated the success of your objectives (from their point of view) you need to do some analysis of what they have said. Any points they have made about features that cause difficulties, or features they have now realised they need, must be included in your suggestions for how the outcomes of your project could be improved. See **table 18** which shows how a candidate might analyse the feedback from **table 17**.

Table 18 – Example of analysis of evaluation

No	Objective	Performance Criteria	Evaluation
1	The end user must be able to create the database when the system is run for the first time after installation	The end user can enter the location of the database file and create the database tables on first run of the program after installation	"I found this feature hard to use initially as it was not creating the database tables. I realised after some trial and error that it was because I wasn't using the correct syntax in the filepath I was inputting. How the filepath should be constructed needs to be made more obvious – especially if we ever have cause to recreate the tables again" How this outcome could be improved? I can see that this is a big problem for the end user if they struggle to get this part right as it prevents the entire system from being of use. I need to add a help feature that explains to the user how to construct the filepath and also I could use a regular expression to validate the filepath entry.

Evaluation: Evidence Checklist

Key to the importance of sections of evidence:

- Any item marked with an **E is essential** and **must be included** if you are to gain good marks in this section
- Any item marked with a **D is desirable** and **should be included,** it will help to gain good marks but is not always essential
- Any item marked with a **C is complementary**. It may help explain aspects of your project but is not essential

Item No	Have you included???	Tick
1	Have you evaluated **your own performance** against your original objectives? **[E]**	☐
2	Have you got some **independent feedback** on your performance against the objectives? **[E]**	☐
3	Have you **analysed that feedback**? **[E]**	☐
4	Have you made **suggestions as to how the outcomes could be improved** (using the feedback gained) **[E]**	☐

Evaluation: Assessment Criteria

There are 4 marks available for this section

Mark band	Advice	Mark range
Upper Mark band	• There is a detailed description of **how well** each objective has been achieved • There is evidence of the gaining, and analysis of the results, of **realistic independent feedback** of the solution • The **feedback has been used** to discuss meaningful ways in which the solution could be improved if the problem that was attempted was revisited	4
Upper Middle Mark band	• There is a detailed description of how well all to nearly all of the objectives have been achieved • There is evidence of the gaining of realistic independent feedback of the solution but there is **limited analysis** of this. • There is **some discussion** of meaningful ways in which the solution could be improved if the problem was revisited	3
Lower Middle Mark band	• There is a description of how well some objectives have been achieved. *(this may be because not all of the objectives were achieved or because they have just been left out)* • There is either no independent feedback or it is so superficial as to be of little use	2
Lower Mark band	• There is a **limited description** of how well some objectives have been achieved. • There is either no independent feedback or it is so superficial as to be of little use	1

Notes

