
Coursework Guidance Booklet

Module 6 - A2 ICT

This booklet is designed to guide you through the process of developing the WRITTEN component of the Module 6 coursework. You should develop an appropriate solution to the problem in conjunction with the written component - as outlined in this booklet.

Use this booklet in conjunction with advice given to you by the examination board and by your teachers. The booklet may appear to make the project look very similar to the Module 3 ICT coursework - do not make the mistake of thinking that you can write the same - you can't. This project is distinctly different, requiring you to develop an Information System and outline the development in a great deal of detail.

ABOUT THE PROJECT - the 'real end user':

You need to find a real end user who has a task you can solve using a computer. The real end user will provide you with a(n):

- task and the requirements you need to meet to fulfil that task;
- point of contact during the project for clarification of questions;
- evaluation of the final completed solution.

The whole project is about the requirements and needs of the **real end user** and this should form the basis for everything you plan to do. In order to get this right, you must:

- start the project finding a *suitable* real end user who will provide a relevant task to test your knowledge of the course;
- interview the real end user extensively at the beginning of the project and base your analysis and design of the solution on the needs the user has stated;
- be absolutely sure you follow this where required.

ABOUT THE PROJECT - from the examination board:

The problem will be of a **substantial** nature and is intended to **integrate** the various skills and concepts developed during the course. The emphasis will be on the project being an **open system of a cyclic nature**, such as being repeated once a year or once an event. The solution is likely to involve the appropriate use of a **range of advanced features and functionalities**. It is possible that these may be provided by a suite of generic application software.

It is not within the spirit of this specification for you to use a stand-alone general purpose programming language.

To gain high marks, it is expected that the candidates' solution must accommodate the **system's information flow and data dynamics**. This might include data flows between packages, such as Dynamic Data Exchange. There is likely to be some consideration of initialising the system, clearing down data from the previous use, processing data, transferring data such as logging transactions and archiving data.

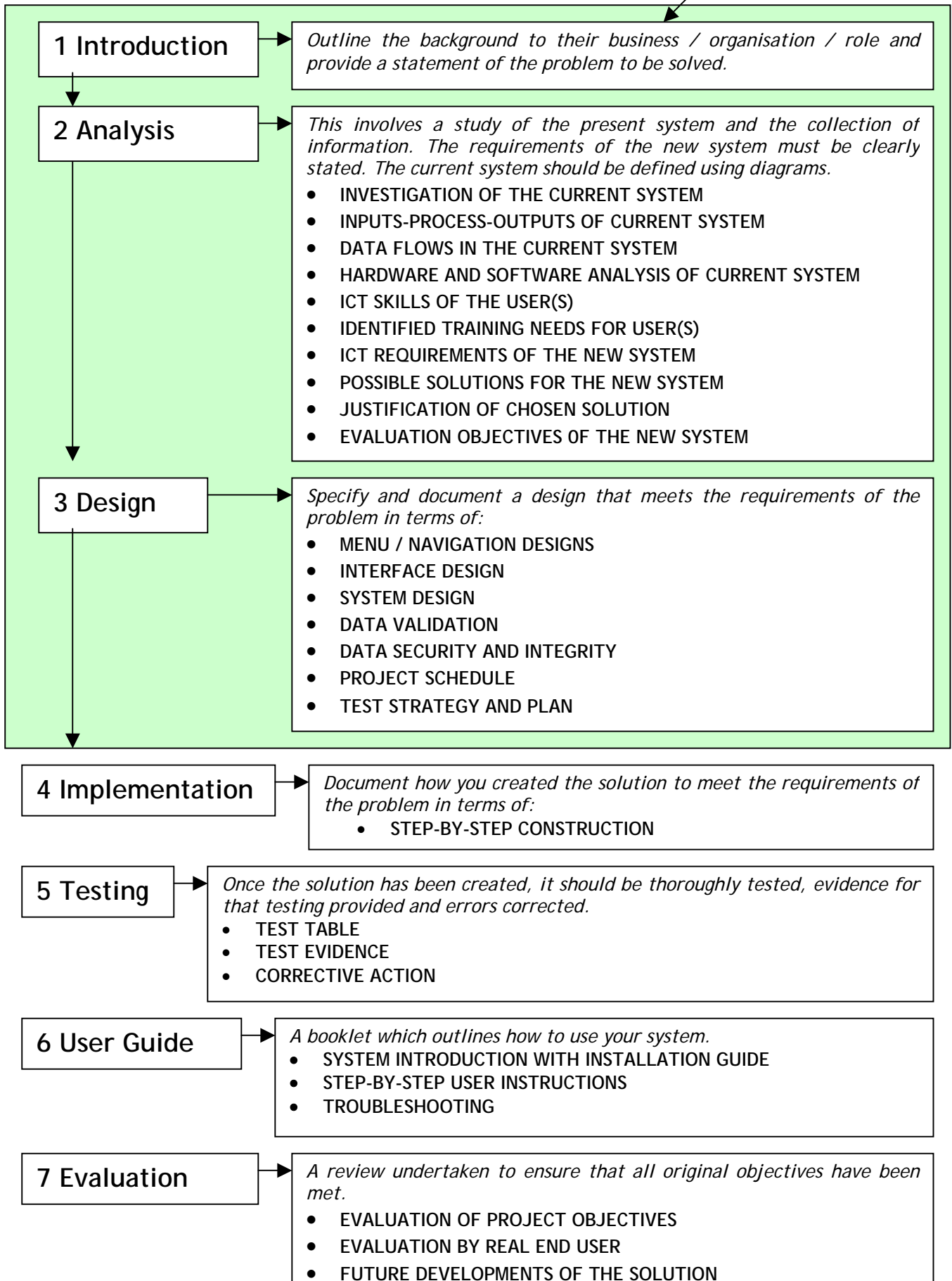
SECTIONS IN COURSEWORK	MARKS
Analysis	18
Design	16
Implementation	15
Testing	15
User Guide	8
Evaluation	10
Report	8
TOTAL	90

You should interview your real end user extensively before you start the project. Look at what you have to do and then make sure you ask the right questions.

The analysis and design sections of the Systems Development Life Cycle **MUST** be completed before you begin work on your solution.

Project Overview

Areas bordered are those that should be completed before the solution is started.



Writing the Project

Each section of the project is outlined below. You should already have a real end user, have had the project approved by your teacher through filling in a coursework proposal form and have carried out information gathering exercises. Gathering details about any system can involve:

- ❑ **Interviewing staff at different levels** - from end-users to senior management;
- ❑ **Examining current business and systems documents and output** - may include current order documents, computer systems procedures and reports used by operations and senior management;
- ❑ **Sending out questionnaires** - the questions have to be carefully constructed to elicit unambiguous answers;
- ❑ **Observation of current procedures** - by spending time in various departments; a time and motion study can show where procedures could be more efficient, or to detect bottlenecks.

For most systems, an extensive interview with the real end user (which is fully documented and included in the appendix of the project) will be sufficient.

1 - Introduction -----

Outline the background to their business / organisation / role and provide a statement of the problem to be solved.

2 - Analysis -----

- **INVESTIGATION OF THE CURRENT SYSTEM**

Here you need to give details about the background to the current system (how does it currently work?).

It is vital that you collect together documentation on the way that the system is currently used in order to demonstrate those problems. This could be a paper invoice that you intend to computerise, etc. These should be **annotated** and included either here or in the appendix and referred to here.

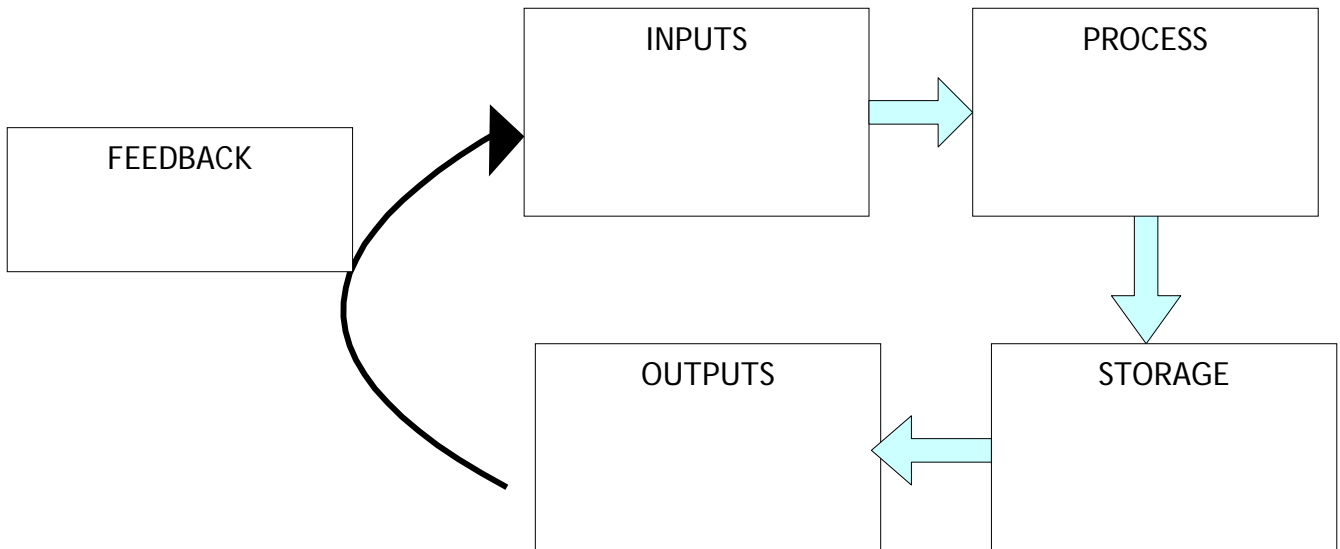
- **INPUT-PROCESS-OUTPUT OF CURRENT SYSTEM**

Draw an IPSO (inputs, processes, storage and outputs) diagram for the current system here, filling in each box with relevant information from your earlier work in design. You may need to add a FEEDBACK arrow to show how any outputs go back into the system. The diagram at the top of the next page shows a structure for this and should be much larger when drawn in the project.

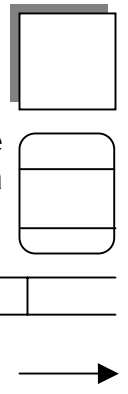
- **DATA FLOWS IN THE CURRENT SYSTEM**

A data flow diagram shows how data moves through a system and what data stores are used. It does not specify what type of data storage is used or how the data is stored. You need to draw one of these to show the current system - use the scenario details you have been given.

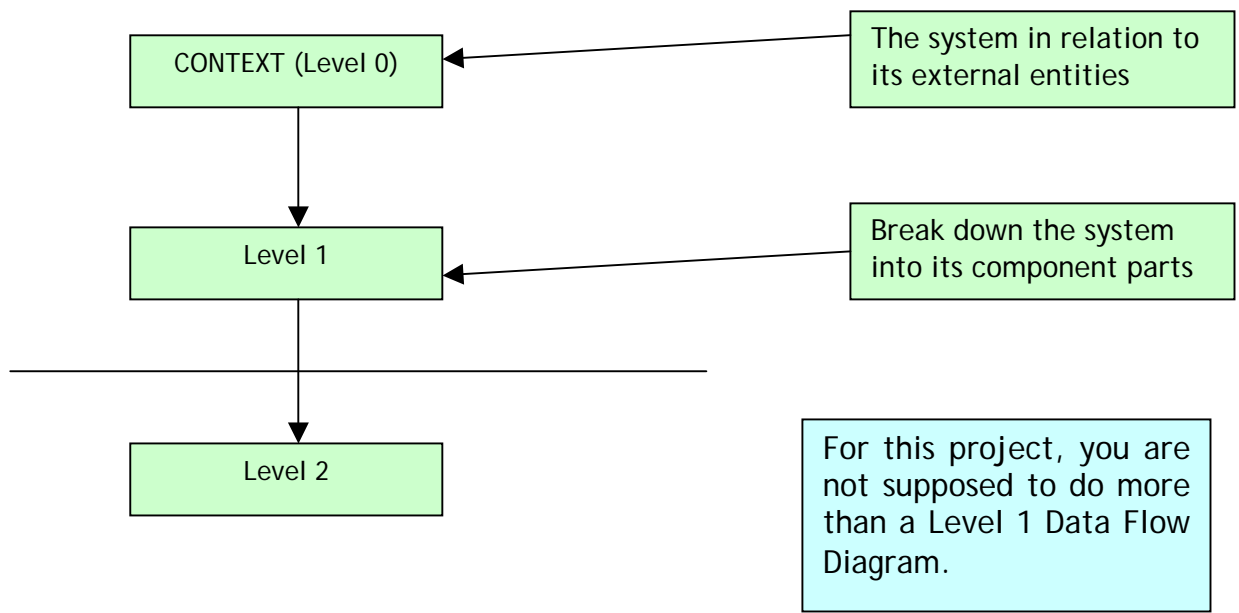
An IPSO Diagram:



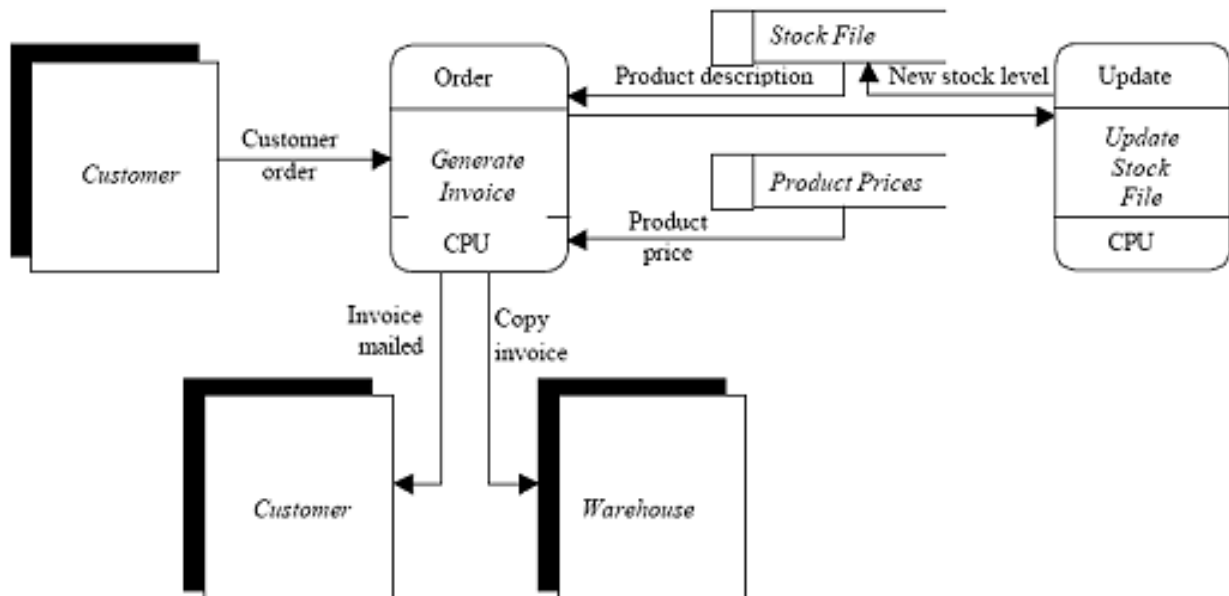
- **External entity - data source or destination** - e.g. people who generate data such as a customer order, or receive information such as an invoice.
- **Process - an operation performed on the data** - the two lines are optional; the top section of the box can be used to label the process, the middle to give a brief explanation, the bottom to say where the process takes place.
Make the first word an active verb - e.g. *validate* data, *adjust* stock level.
- **Data store** - such as a file held on disk or a batch of documents.
- **Data flow** - movement between entities - processes or data stores
Allowed dataflow connections are between:
 - Process and data store
 - Process and external entity
 - Process and process



Data Flow Diagrams are broken down into several layers:



An example of a data flow diagram (LEVEL 1)



- **HARDWARE AND SOFTWARE ANALYSIS OF CURRENT SYSTEM**

Carry out a hardware and software analysis for the real end user's computer system - the one the solution will end up on.

HARDWARE: include brief specifications of the hardware for the computer system used;
SOFTWARE: list the relevant software on the system (not everything, but anything relevant to the scope of the project).

At the end of the section, make brief conclusions about:

- the possible software you could use and
- any potential problems you can foresee with the specifications of the computer systems you will be putting your solution on.

- **ICT SKILLS OF THE USER(S) / IDENTIFIED TRAINING NEEDS FOR USER(S)**

Referring to the questionnaire you undertook at the beginning of the project with your real end user, you should outline what skills the ICT user(s) has / have, in this section. Bullet point a list of their experiences and identify areas in which the user(s) may require training.

- **ICT REQUIREMENTS OF THE NEW SYSTEM**

This needs to be a detailed section looking carefully at the system you are going to create. You MUST refer to the questionnaire in order to show what the real end user has requested in terms of the general system. You cannot decide what software you will be using yet - that comes in the next section.

- **POSSIBLE SOLUTIONS FOR THE NEW SYSTEM**

List the potential non-computerised and generic package solutions you could use to develop your system. Make sure you:

- Indicate why each one would be better than the existing system;
- address the strengths and weaknesses of each product or method.

Including possible non-computerised solutions (such as a paper-based solution) to the problem set is important as it shows you understand that computers are only one possible tool amongst many. Obviously you cannot choose it, as this is an ICT project after all!

It is not a good idea to be doing a project which does not involve either a database or spreadsheet package as this is a project requiring you to make an Information System!

- **JUSTIFICATION OF CHOSEN SOLUTION**

Once you have chosen the solution you would like to use to create the system, you must justify why you think it is more effective than any other method. **MAKE SURE THAT YOU LINK THE JUSTIFICATION TO THE INFORMATION IN THE HARDWARE SECTION.** The package you choose to work in must be on the system it will end up on, etc.!

Overall, it is vital that you think about the users in all of this. Is the solution the best for them and their system?

- **EVALUATION OBJECTIVES OF THE PROJECT**

You are expected to set three types of objectives against which you will measure the success of your project:

- General system objectives
- Qualitative objectives
- Quantitative objectives

These will be useful to steer you in the right direction during your implementation and later, will be used as a basis for a review of the project in the evaluation section at the end of the project.

Remember the following points:

- *to set your objectives directly from the interview of your real end user.*
- *when you write out objectives, set them out as statements with bullet points. Do not explain them in detail.*

General system objectives - in this section of your objectives you should make general statements about how you expect your whole system to work:

1.8 System Objectives

1.8a Introduction And General Objectives

Taking into account all the research into the background of the problem and visions that the manager has of a new system for managing and controlling stock information, and report creation for Weybridge Post Office the following two section detail qualitative and quantitative objectives that the system must meet to be deemed successful. First of all the general objectives are stated.

- The system must adhere to the workings of the old system whilst computerising as much as possible.
- A point and click easy to use GUI should be used with the option of using the keyboard. In addition the interface should be uniform throughout, appealing and subdued.
- The system should be as automatic as possible, reducing the need for user intervention where possible.
- The system must be stable and reliable; it must not breakdown or crash which may result in lost data.
- The system should be secure. Unauthorised personnel must not be able to gain access to the details stored on the system.

Qualitative Objectives:

Qualitative objectives are statements that lay out the more 'open-ended' aspects of the project. They are **not** measured by specific timings or amounts (which is done by the quantitative objectives). They can be more *subjective*: see the example on the next page.

Qualitative: "relating to or involving quality or kind."

Quantitative Objectives:

The key difference between qualitative and quantitative objectives is the term "*measurement*". For example, a qualitative objective is "*to design the system to hold records of customer orders*". This is difficult to prove directly. Quantitative objectives should include numbers or measurements which allow them to be directly measurable, such as "*the system should be able to store the details of over 1000 customers*". Quantitative objectives are more *objective* than *subjective*.

Quantitative: "relating to or involving the measurement of quantity or amount."

Qualitative Objectives Example:

1.8b Qualitative Objectives

- The system should store details of all products and suppliers.
- The level of stock of any product should be quickly and easily identified, thus allowing the user to account for every item of product.
- The system should eliminate the need for the user to travel between different locations to determine the level of stock for all products.
- The system should automatically calculate the level of stock of each product after deliveries and after sales updating.
- Products that need reordering should be easily identified perhaps using a report facility.
- The system should allow the user to gain accurate information regarding past sales figures, profits and revenues, easily and efficiently.
- The system should be able to verify values of sales derived from counting stock with the value from the cash register.
- The user should be able to enter all invoice information and view it at a later date.
- All the above should be clear and easy to do.
- So not to affect the running of the business, all calculations must have accurate and true results.
- The system should have an attractive unobtrusive interface.
- Every part of the system must be accessible to the user with minimum fuss.

Quantitative Objectives Example:

1.8c Quantitative Objectives

- The system should be able to store details for 1500+ products and 10+ suppliers.
- The system should be able to hold at least two years worth of sales figures for individual products, and at least 18 months worth of invoices.
- The current level of stock for any product should be able to be identified within 30 seconds of starting the program.
- The items needing to be reordered from a particular supplier should be identifiable within 30 seconds of starting the program.
- Past sales figures and invoice details should be identifiable within 30 seconds of starting the program.
- The user must be using the system within one week of receiving it without further help.

3 - Design -----

- MENU / NAVIGATION DESIGNS and INTERFACE DESIGN

In this important section, the focus should be on the Human-Computer Interface of your solution. In order to show this, you should design and draw out the following:

- ❑ a menu hierarchy diagram;
- ❑ input and output screen / form / report designs.

All interface designs can be shown in one of two possible ways:

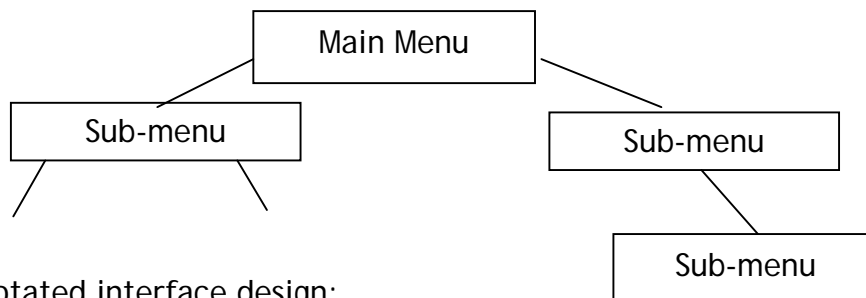
- ❑ as hand drawn designs;
- ❑ or as **prototyped** screenshots.

NEVER use real screenshots from your implementation to show your design! You should not yet have started your solution.

It is vital that all designs are ANNOTATED to explain the following:

- ❑ the reasons for your designs in terms of HCI;
- ❑ how you will try to meet the requirements of your real end user;
- ❑ what advanced skills you will need to learn to put the project together in the generic package(s) you have chosen.

An example of a menu hierarchy diagram:



An example of annotated interface design:

Company Name is clearly visible at the top. This is to show ownership of the solution. It also stops this exact solution being copied for another business.

The exit solution button. This will exit access and return the user to windows. There will not be a "do u wish to save option".

Tods Tyres

- Add Customer
- Existing Customer

Created by Dynamic Computer Solutions
Website : Dynamic-Computer-Solutions.co.uk
Email : Dynamic@ComputerSolutions.co.uk

This is just telling the user who created the solution and how to contact them if need be.

This is where the main options will appear. This is to allow for easy navigation of the features. The font is a clear and simple one to help ensure the user clicks where they mean to. The button size will be the size of the words. Also called the main display area. The two buttons here, once clicked will lead through to more indepth menus.

- **SYSTEM DESIGN**

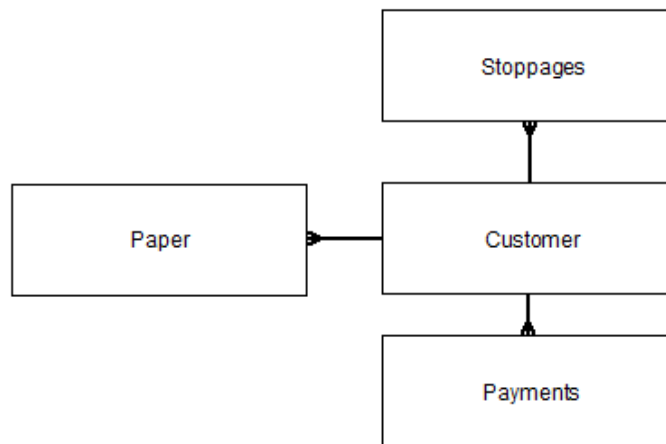
There are four key sections to cover here:

1. **Data Flow Diagrams (DfD)**

Use the information from the Analysis section to help you draw a DfD for the new system you are designing.

2. **Entity-Relationship Diagrams and Normalisation**

If using a database package, it is important that you show the relationships between the entities in your system by drawing an entity-relationship diagram. You will know how to do this already, so just remember that all entities must be joined - don't draw each relationship separately! For example:



For normalisation - you will need to put the data into THIRD NORMAL FORM (3NF). Use notes from the course to carry out normalisation (use your Heathcote textbook).

Don't forget to use the standard notation for describing tables in a relational database when showing HOW you normalised the data.

3. **Systems Flowchart**

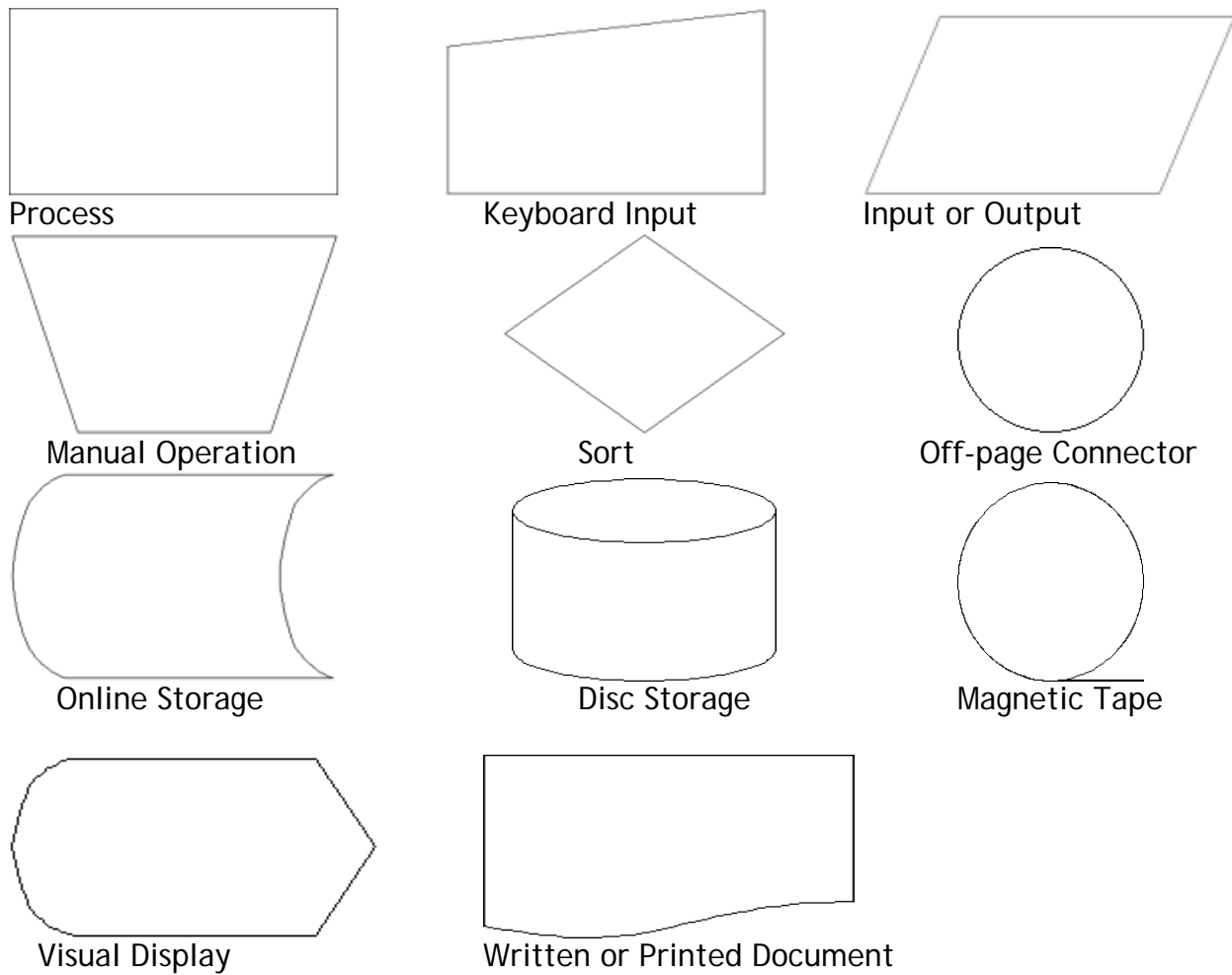
A systems flowchart is a diagram showing an overview of a complete system. This kind of diagram shows:

- tasks to be carried out in the new system, whether manual or by the computer.
- devices (disc drives, tape drives, terminals, etc.) that are to be used in the system.
- media used for input, storage and output.
- files used by the system.

Drawing a Systems Flowchart - there are many factors to be taken into account when designing a new system. You must establish the following facts:

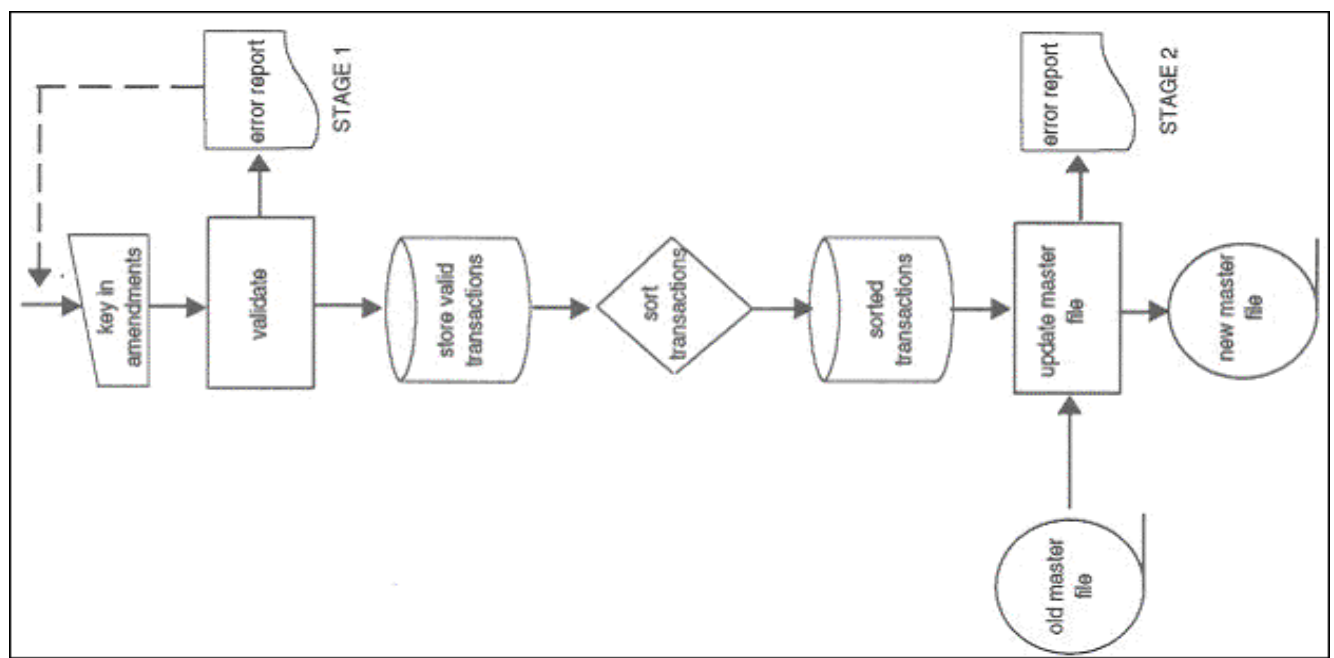
- ❑ how is data to be captured and input into the system?
- ❑ what storage media will be used: disc, tape or other?
- ❑ what will be the output from the system?
- ❑ what are the processing steps to be carried out?

Systems Flowchart Symbols:



A Systems Flowchart Example:

NB - This diagram is turned on its side to get it into the booklet!



4. MACRO DESIGN

In this final section of 'Systems Design' you should think about the macros you will be using in your project. You need to outline what each macro will be doing. You can present your macro designs in one of two ways:

- As flowcharts showing each of the steps a macro will take to do its task;
- As a bullet pointed list showing the same steps.

• DATA VALIDATION

Outline the specific validation methods you are going to use. You need to draw out a table in the style of the following example in order for you to show more detail about the validation techniques you are going to use:

Field(s)	Validation	Method
Car registration number	Must be in the LL 00 LLL format	Program using a validation technique / input mask o validate the entry of data into the system.

• DATA SECURITY AND INTEGRITY

There are two key sections here to address:

- **SECURITY OF DATA** - you must address the protection of the data on the system in the event of a disaster (natural or otherwise). How will data be restored? Look at a BACK-UP methods and storage.
- **INTEGRITY OF DATA** - look at how the *correctness* of the data will be checked / designed into the system. Show how you will ensure that the data that is entered into your system will be faithful to what it is trying to represent (e.g. how the system keeps a customer's telephone number for a customer).

You will also need to address two issues with regard to SYSTEM SECURITY:

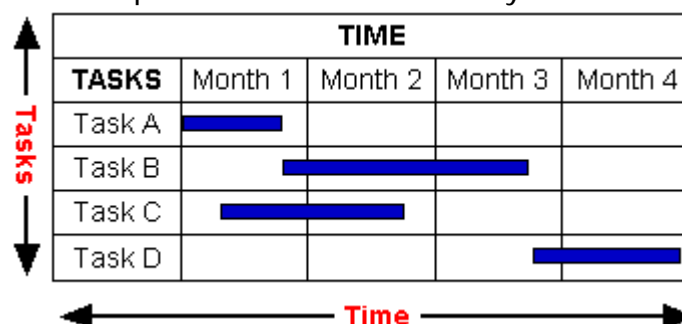
- **PHYSICAL SECURITY** - look at the security of the area around where the computer will be situated (e.g. the room it is in, etc.);
- **ACCESS CONTROL** - how the solution you have created and its data will be protected ON THE COMPUTER(S). What about passwords and if someone forgets them, for example?

• PROJECT SCHEDULE

You need to carry out two tasks here:

- Break down the whole solution into sub-tasks;
- Schedule the tasks into the remaining time you have left.

The most effective way to do this is to draw a Gantt Chart. Gantt charts are a project planning tool that can be used to represent the timing of tasks required to complete a project. Gantt charts are simple to understand and easy to construct:



- **TEST STRATEGY AND PLAN**

The test strategy section gives you an opportunity to design the tests you intend to perform on the solution you have designed. It is important to outline your OVERALL STRATEGY which looks at how you will go about your testing:

- o Are you going to test all procedures and functions?
- o If your project does not have any existing real data to use as test data then where are you going to get your test data?
- o If you are making up your data then how will you justify it?
- o Above all, how are you going to test your system in such a way as to find as many potential problems as possible?
- o How are you going to test your validation routines?
- o Etc.

The following areas then need including:

- **Reason for testing** - examiners would like a brief outline of why you are testing your project. This is so they can see that you understand why you are doing it. The following are possible reasons:
 - o the purpose of testing is to find and iron out errors;
 - o to check that all parts of your system work, no matter what data is input;
 - o to check that all parts of the system as originally specified are present.
- **Test plan** - a test plan takes the form of a **table** listing the tests to perform, It should cover every possible type of input. The test plan needs to show for each test what the expected result should be.

Test plans MUST include the actual data you intend to put into your system in the test plan and which kind of data it is must be stated:

- o **TYPICAL** - data you would expect to be put into your system;
- o **EXTREME** (boundary or awkward) - data on the edges of the range of typical data you would expect to put into your system;
- o **ERRONEOUS** - data which is clearly full of errors or that would not be expected to be accepted by your system.

The test plan should be drawn in the following structure:

Test Number	Test Description	Test Data	Expected Outcome
T14	<i>Make sure that the input form for the Exports Table accepts product names</i>	<i>BANANA (typical)</i>	<i>Accepted and saved</i>
T15		<i>BRAZIL NUTS (boundary)</i>	<i>Validation box appears to check entry</i>
T16		<i>XXXXCV (extreme)</i>	<i>Not accepted</i>

BEWARE! Simply taking screenshots and putting them straight into your document will create a huge file size, which you may find difficult to transfer between school and home. In order to avoid this follow this step-by-step guide to reducing file size:

- ❑ take the screenshot using the 'PRTSRN' button;
- ❑ open Paint Shop Pro and PASTE the screenshot into the program;
- ❑ reduce the size of the screenshot using the IMAGE - RESIZE command;
- ❑ then save the image as a '.jpg' from the FILE - SAVE AS command;
- ❑ go into your Word document and then INSERT - PICTURE - FROM FILE.

5 - Testing -----

• TEST TABLE

Transfer the test plan you devised in the design section and add two columns as shown below. You can now carry out the tests as planned.

Test Number	Test Description	Test Data	Expected Outcome	Actual Outcome	Corrective Action
T14	Make sure that the input form for the Exports Table accepts product names	BANANA (typical)	Accepted and saved	Failed - it did not save	See CA4
T15		BRAZIL NUTS (boundary)	Validation box appears to check entry	Passed	
T16		XXXXCV (extreme)	Not accepted	Passed - it did not accept extreme data	

• TEST EVIDENCE

In this section you should include evidence (in the form of screenshots) for the tests that are **successful**. You should ensure that:

- ❑ you **sample** the screenshots. You should not include evidence for every test, but every TYPE of test. So if you are testing 10 buttons, only evidence ONE button test and then cross-reference to the test numbers of the others;
- ❑ you **annotate** the screenshots. Show the test results, indicate the test data, its type and anything else relevant to the test;
- ❑ your screenshots are in-situ.

• CORRECTIVE ACTION

You have to have some corrective action as a result of a test failure. In this section, you should include annotated screenshots showing:

- ❑ the failure;
- ❑ how you put the failure right;
- ❑ the resulting test pass.

Again, make sure that your screenshots are in situ.

6 - User Guide -----

Your manual should be a 'stand alone' document designed for your real end user. Make sure that it is at a level appropriate to the actual user and include realistic operational details appropriate to the actual situation. When you have used a generic application package, DO NOT give an explanation of how that works - you must only give information on **your system!** You must include the following three sections:

- **INTRODUCTION TO THE SYSTEM**

Include a contents page, then start the section with an introduction describing the overall function of the system and an **installation guide**.

- **STEP-BY-STEP USER INSTRUCTIONS**

Include the step-by-step guide to using the system. Try to vary what you show in the sample. Include screenshots with annotations and most importantly, *DATA IN SITU!* Good explanations are needed with the screenshots - the annotations do NOT replace this.

- **TROUBLESHOOTING**

Explain in a suitable format (such as a FAQ or list of possible errors), how to deal with possible error messages that you may have built into your solution and how to deal with possible problems. Make sure one of your entries addresses the question of how the user should **back-up** their solution. This then covers back-up without having to have a separate section for it!

7 - Evaluation -----

- **EVALUATION OF PROJECT OBJECTIVES**

When you evaluate, you are measuring and reviewing your progress against objectives. In order to do that in this project, you should copy and paste all of your original evaluation objectives into this section and address each one in turn.

The nature of this kind of project means that you **will not have** achieved perfection - therefore a realistic evaluation means that you should find what worked and what didn't. Do this by isolating each objective and discussing what happened.

You must be specific - vague responses will be penalised.

Be careful in the evaluation - this isn't an evaluation of your performance or how hard you found the project or how hard you found sticking to the deadlines. No-one is interested in this. **Please stick to evaluating your solution.**

- **EVALUATION BY REAL END USER**

At some point **BEFORE NOW** you will need to have given your real end user the completed solution and user manual in order that they can evaluate the system. They should know what they originally asked for, so should comment on how well you have achieved those requests. An example of a format for this is on the next page.

It is important they suggest ways of improving things. It would be far too unlikely in the real world that you would ever have got it perfectly right first time - look at Microsoft!

An example of a real end user evaluation letter can be found over the page:

Evaluated Objective Examples:

Evaluation

6.1 Comparing System Implemented With Project Objectives

6.1a Qualitative Objectives

The following section compares the implemented stock control system, with the qualitative objectives the system was to fulfil, set out in the Analysis section of this report.

The system should store details of all products and suppliers.

This objective has been completely fulfilled. Details of each supplier and every product the business sells can be stored on the system, with tblSuppliers and tblProduct being the data stores. Details are easily viewable as well, with the New Supplier and New Product forms able to check details.

The level of stock of any product should be quickly and easily identified, thus allowing the user to account for every item of product.

This has been accomplished through the use of frmViewStockLevels. The user can now quickly navigate to the form, and select the category or supplier he wishes to view the stock levels of. This will be mainly used if there are any stock queries, however the fact products can be filtered by their supplier or category is also of use. For instance, if the user is contemplating whether to create an order for a particular supplier, he can use this form to get an overall view of products from this supplier. In addition after updating stock levels by category, they can be viewed easily by category.

• FUTURE DEVELOPMENTS OF THE SOLUTION

List the improvements you could make and any possible developments you would suggest adding to the solution in future. *It would be astute to include any suggestions made by your real end user in their evaluation form!*

25th April 2002

Dear

As you know, we are really impressed with the program that you have made, and have used it several times since it was installed in the shop. First of all I was struck by how professional it looked, but also found it was easy to move around in. In many ways it is similar in the way it is organised, to the Post Office Horizon system, with each switchboard acting as a folder.

When the system was first set up it took a very long time to enter all product details, especially as each cost per unit had to be calculated. As you know it had to be done after closing, and took about four days to get the system up to date. After this was done though, the system has been very useful. Two orders were created on the first week, which saved a lot of time. Gradually, the time initially spent will pay off in time saved.

I can think of a few improvements however. The first concerns product delivery. When items are not delivered, there is no way to account for this. Only the whole order can be confirmed, there should be a way to confirm individual items. There should be a way to view a sales log of only one product, also. At the moment viewing a whole category's sales figures can be a waste of time.

Lastly it has occurred to me that there is no way to work out VAT values from the program. The program thinks that all items have the same VAT category, but this is not true. Some items are zero rated, for instance Milk. As VAT returns are made every 3 months, it would be very useful if it could be incorporated into the program.

Thanks

Example of a real end user evaluation letter. Use this to show what the real end user likes and what the real end user would like to see in the future.

End of Project!

Project Checklist

Tick off each section as you do them or as a final check:

MAIN SECTION	SUB-SECTION	✓
INTRODUCTION	INTRODUCTION	
ANALYSIS	INVESTIGATION OF THE CURRENT SYSTEM	
	INPUT-PROCESS-OUTPUT OF CURRENT SYSTEM	
	DATA FLOWS IN THE CURRENT SYSTEM	
	HARDWARE AND SOFTWARE ANALYSIS OF CURRENT SYSTEM	
	ICT SKILLS OF THE USER(S)	
	IDENTIFIED TRAINING NEEDS FOR USER(S)	
	ICT REQUIREMENTS OF THE NEW SYSTEM	
	POSSIBLE SOLUTIONS FOR THE NEW SYSTEM	
	JUSTIFICATION OF CHOSEN SOLUTION	
	EVALUATION OBJECTIVES OF THE NEW SYSTEM	
DESIGN	MENU / NAVIGATION DESIGNS	
	INTERFACE DESIGNS	
	SYSTEM DESIGN	
	DATA VALIDATION	
	DATA SECURITY AND INTEGRITY	
	PROJECT SCHEDULE	
	TEST STRATEGY AND PLAN	
IMPLEMENTATION	STEP-BY-STEP CONSTRUCTION	
TESTING	TEST TABLE	
	TEST EVIDENCE	
	CORRECTIVE ACTION	
USER GUIDE	INTRODUCTION TO THE SYSTEM	
	STEP-BY-STEP USER INSTRUCTIONS	
	TROUBLESHOOTING	
EVALUATION	EVALUATION OF PROJECT OBJECTIVES	
	EVALUATION BY REAL END USER	
	FUTURE DEVELOPMENTS OF THE SOLUTION	