Unit 3 – Programming Project

3.1 Analysis of the problem

3.1.1 Problem Identification		
Task	**Description**	**Completed? (✔)**
Background of the problem area	• Introduce the person/organisation you are designing the solution. Paint the picture for the examiner.	
Problem Area /Problem Identification	• Describe the problem area that you will investigate	
Proposed Solution	• What solution do you recommend? • Describe how you think the problem can be broken down into areas	
A computational Approach	• Explain why the problem is amenable to a computational approach. • Describe and justify the features that make t he problem solvable by computational meth ods.	
3.1.2 Stakeholders		
Stakeholders	• List specific names, job titles, groups of people or target audience of people who will be using your solution • Describe each of these stakeholders in detail • Describe how these stakeholders will be interested in the solution you create/How they will make use of the solution	
3.1.3 Research the problem		
Investigation Plan	• This should include a list of fact finding techniques and advantages and disadvantages of you using each one.	
Use fact finding techniques	• Iterative process – Use a number of fact finding techniques to collect information about the current system/new system • Make sure to include the following for each one: - What technique you used and why you are using it e.g. questionnaires and what you hope to find out by collecting this information - Who was involved in the fact finding technique and when did it take place e.g. end users (try to use names)	

	- Any screenshots of evidence of you setting up this fact finding technique with end user e.g. emails - Any screenshots/documents of notes/questionnaires collected e.g. interview transcript, template of questionnaire	
Analyse results of fact finding technique used	● For each fact finding technique you must analyse your results e.g. what did you find out? Could this be put into some kind of pie chart	
Existing Solutions	● Find several existing solutions are that similar to the solution you will be creating. Discuss the following: - What is the solution - What are the advantages of this solution? What did you like about it? - What are the disadvantages of the solution? What could be improved? What might be a major flaw in the system? - What features of this system could you use in your system?	
Essential Features of the solution	● You need to describe the essential features of your system - Each one must be explained (you must explain your reasoning for the choice of essential feature – why is it needed)	
Limitations of the solution	● Explain the limitations of the proposed solution. ● Can you identify two or three things that you could do, that perhaps your user has said would be good if you could, but you are not going to? ● These are limitations (and they can also be things that you can suggest for the future). ● Why are you not going to do them? Is it because of time problems? Is it because you have drawn up a list (with evidence) of the most important things, and then a list of other things if there is time? Is it because you don't have access to the technology e.g. the user wanted to scan in something using biometrics? Is it because you don't have the necessary skills to connect your proposed solution to the customer's database? ● There are all kinds of things that you potentially could do, but aren't going to!	
Requirements Specification (solution requirements)	● Specify and justify the solution requirements - This should be broken up into input, output, process and validation requirements.	

	- For each requirement you must justify why this is a requirement for your solution e.g. need evidence refer back to investigation https://www.computerscience.gcse.guru/theory/validation	
Requirements specification (hardware requirements)	• Specify and justify any hardware and software requirements needed for your solution	

Success Criteria	● Identify and justify measurable success criteria for the proposed solution. - The success criteria should focus on how you will meet each requirements - You need to be able to demonstrate that you have met this part of success criteria through testing therefore making it measurable - This could be done in a table (success criteria in one column and details of how you might test that success criteria to show that you have met it) - Justification	

Design of the solution		
Task	Description	Completed? (✔)
Decompose the problem	● Break down the problem into smaller parts suitable for computational solutions justifying any decisions made. ● Look at the Requirements Specification. Write a list of the jobs that need doing to meet all of the requirements. ● Can any of the jobs be broken down even further? Try to break down jobs into units so that each unit is for one identifiable task. ● Describe in more detail what is involved with each of the jobs in your list. ● Try to justify why you have broken the problem down into the component parts on your list. For example: ***You might decide that you need to create classes for each of the main characters in a game but because they all share some characteristics, you will design a superclass called 'Character'. Using inheritance will simplify the overall design.*** ***You might have to produce an ordered list of items to display so you will use a standard sorting algoithm to do this rather than write a completely new one. You will do this to save time.*** ***You might have two characters fighting in a game so you will produce a function that receives two characters, lets them fight and then outputs the result. You will do this because this can then be used for all fights between two characters.*** ***You need to use a lot of different sounds in your game. You will therefore use an open source library of sounds so you don't have copyright issues.***	

Data Structure Design (Variables and Arrays (Lists))	You will need to include a table with all the variables that you plan to use in program. It should include the following: • Identifier name • Data type • Description of the variable • An example of data stored in the variable • Size of the variable (refer to your data type sizes table) • Any validation you plan to carry out with this variable • Justification of why this variable is needed **You will also need to do this for any arrays that you have** **You will also need a table that includes a list of files that your program will use. This should include:** • File name • Description of the file • The data type of the data it will store • Size of each record • How many records each file will have • Sample record that will be stored in each file Underneath this table, you should include the file size calculations.	
Pseudo Code and Flowchart	You need to include a number of algorithms for each screen / important algorithms. Remember to include line numbers	
Design Layout	• In this section you have create a storyboard. • You must include the layout of each screen. • This can be done on paper and then scanned to be pasted in your document or you can use PowerPoint to create each layout.	
Testing Strategy	You need to include a test plan for each screen. You should include the following: • Test Number • Test Description • Test Data to be used • Test Type • Expected Result	

Developing the solution		
Task	Description	Completed? (✔)
Iterative Development Process	• Provide annotated evidence of each stage of the iterative development process justifying any decision made. • Provide annotated evidence of prototype solutions justifying any decision made. **a)** Put a heading, ***3.3.1a Stages of development***. **b)** You should provide annotated screendumps of each stage of the development of the project. **c)** This isn't actually a very difficult task if you do it as you go along. The trick is to know how to take a screenshot of just a portion of your screen. (Your teacher will show you for whatever operating system you are using, or if you are using Windows, open the Snipping tool and use that.) You also should create a folder and just dump each screenshot in it, using the date in the file name. **d)** As you do each screen dump, add comments to them. For example, you should describe what a particular screendump is showing. Then you should explain what you have done since the last screendump, and say why you have done that. **e)** How many screendumps do you need? Your teacher will guide you but you should have a screendump for each major area of development. **f)** Put a heading, ***3.3.1b Prototype solutions***. **g)** Part of the process of development will be to make prototypes. These are simplified versions of either a part of the project all the whole project. The purpose of them can be for your own benefit, to try out different approaches in code or to check you really can use a technique in the programming language you have selected. You might even put together a very simple version of a project before deciding what project to do, to check that it is a feasible project. A prototype could be to show a customer what each screen in a game will look like and the data and buttons on each screen, so one prototype might just be a series of screens in PowerPoint that you put together in half an hour. All of these are valuable evidence of prototype development. You simply have to print them off, describe what each one is showing, why you made it and how it helped you.	

3.4 Evaluation		
Task	Description	Completed? (✔)
Testing to inform evaluation	a) Provide annotated evidence of testing the solution of robustness at the end of the development process. b) Provide annotated evidence of usability testing (user feedback). **Things to consider doing:** **a)** Put a heading, ***3.4.1a Robustness***. **b)** This section is about you or your testers trying very hard to 'break' your software. You should play the Devil's Advocate and try and do anything you can think of to make the software work in unexpected ways or to stop it working properly. This is 'robustness testing' and you may have already included some robustness tests in Section 3.2.3. **c)** The simplest way of doing this would be to draw up a table, list the tests you are going to do, why you are going to do that test, what data you will use if any, what you expect the outcome to be and what the result was when you carried out the test. **d)** Examples of the kinds of robustness testing you could do include: ***Entering in completely random strings when asked for data, including control keys and symbols.*** ***Reading or writing to a file that doesn't exist.*** ***Reading or writing to the cloud without an Internet connection*** **e)** Put a heading, ***3.4.1b User feedback***. **f)** You should carry out the tests you said you were going to do in Section 3.2.3.	
Success of the solution	Use the test evidence from the development and post development process to evaluate the solution against the **success criteria** from the analysis. **Things to consider doing:** **a)** Put a heading, ***3.4.2a Evaluation of the solution***. **b)** You now have to go through your Requirements Specification. Where is the evidence that you have measured the success of each item in the list? You should refer to where it can be found. **c)** Now you have to comment on the degree of success. If the evidence shows that you met a particular requirement fully, then say that. If you only partially completed a requirement or didn't do it all, you need to explain why that was the case. Was it because you ran out of time and the root of that was that your project was unrealistically large and ambitious? Was it that you couldn't implement something because it was too difficult? Perhaps your school's Internet connection was so poor that you couldn't test something fully that needed the Internet. Maybe your user changed their mind about what they wanted half way through the project. **d)** You should try to provide supporting evidence for any reasons you give, e.g. a signed and dated statement from the user saying they asked you not to include XXX after all but to include YYY instead.	
Describe the final product	Provide annotated evidence of the useability features from the design, commenting on their effectiveness. **Things to consider doing:** **a)** Put a heading, ***3.4.3a Useability features***. **b)** You need to describe the final product.	

	c) You could start this section by including a screendump of the of the useability features at the design stage, and then a screendump of the the same features actually implemented. **d)** You could then evaluate the effectiveness of the implemented designs by describing your user using them, what features they can use easily, how they found using the features, what they liked and disliked and so on. You may already have the evidence for this as part of your User Tests and Feedback so you could simply refer to them. If not, then you could draw up a table of tasks you would like the user to do, and then observe them doing them, writing down any comments they make. Then get them to sign and date it.	
Maintenance and development	**a)** Discuss the maintainability of the solution. **b)** Discuss potential further development of the solution. **Things to consider doing:** **a)** Put a heading, ***3.4.4a Maintainability of the solution***. **b)** You need to discuss how the system will be maintained in the future. ***You could start by saying in a sentence what maintenance is, the purpose of it. Don't write an essay on the different kinds of maintenance, however, just that fact that changes may need to be made in the future by people other than the original designers and for a variety of reasons.*** ***You could start by saying that you have created (or would create) a complete, up-to-date and organised set of design and testing documents so that any person who has to make modifications in the future will be able to quickly get to grips with the overall structure as well as the structure of the detailed parts. This is simply a matter of collecting together all of your design documents, putting them in an order and adding an index page and a title page. You teacher should guide you on whether you should actually create this technical document.*** ***You should include a description of any routine maintenance that should be done, what needs to be done, why it needs to be done and how often.*** **c)** Put a heading, ***3.4.4b Further development***. **d)** What is it that your system cannot do now, that if it could, would add something to the system? Are there features that don't exist yet, that if they did exist would make a user's life easier, or make the product more fun to use, or would improve the speed that the system runs at. These are potential areas for development. Examples might include: ***Developing a system of contextual HELP, so that a user could e.g. press the F1 key and the system would cleverly know what screen he was on or where the cursor was pointing, and then pop up with help for that place.*** ***Perhaps the development of some music or sounds would make that interactive adventure game a lot more engaging for younger students.*** ***The inclusion of graphics for a text-based adventure game might be worth exploring.*** ***Being able to play against another person across the Interent would broaden its appeal.***	

	e) Whenever you make a suggestion for potential further development, you need to justify why this might be a good idea and possibly what (broadly speaking) would be involved in implementing it.	