|  |  |
| --- | --- |
| **Analysis** | |
| A description of the problem | 1. Provide an outline of what the problem is. 2. Explain the features required in a computer system that would help solve the problem. |
| Identify all the stakeholders | 1. Identify all the stakeholders that will influence the development of the program. E.g. a teaching program will include teachers and students, a network utility program will include the network manager and the users, for a mobile app you would include a description of the type of group the program is being designed for but choose a few people to represent that group.   \*\* During the whole coursework, you have to provide real evidence of you involving all these stakeholders (through emails, letters etc.) at every stage of the software development process. This has to be from **3 or more people**. |
| Justify why the problem can be solved by computational methods | 1. Explain why the problem is suited to a computer program. 2. Explain the features of the problem that are suitable for a computer program. 3. Explain why the output produced by the program is valuable to the stakeholders involved. |
| Interview | 1. Plan and justify the questions you intend on asking the stakeholders to find out more about the problem and exactly what they require from the new program you are developing (you might have a different set of questions for different stakeholders). 2. Conduct an investigation (either interviews, questionnaires, observations, focus groups or any other appropriate method) with your stakeholders. How you do this is up to you, but justify your approach. Remember some methods are better than others for acquiring detailed responses. 3. Analyse the results of the findings and draw out key information that has come out from the investigation. 4. Conduct further interviews to ask follow questions. Make sure you know exactly what the client wants! (Functionality, appearance, support etc.)   \*\*remember to evidence your client involvement. |
| Researching existing similar problems | 1. Conduct detailed research into what has already been developed/existing solutions. 2. Show which areas of the research you can adapt and apply to your problem. 3. Use the findings of the research to justify the approach that you will be using to create your program (programming language, programming features, GUI, additional features you have decided to add etc.). 4. Identify any limitations of the solution you are proposing. You will need to explain and justify these limitations. |
| Further meeting with stakeholders needed? | 1. You might choose to go back to your client now that you have conducted some research to inform them of any new ideas/limitations. 2. Analyse the results of the research and take note of key information from the meeting.   \*\*remember that keeping stakeholders is very important every step of the way. Remember to evidence your client involvement. |
| Abstraction diagram | 1. Now that you have a better idea of what your client wants, you will need to create an abstraction diagram of the problem. 2. This will need to have a thorough explanation. |
| Features of the proposed solution – stakeholder requirements | 1. Identify all the features to be included in the final proposed solution, these will be known as the stakeholder requirements 2. Identify any limitations of the proposed solution (for e.g. there may be some features that could be added to enhance the functionality/usability of the program if you had further time for this project).   BE REALISITIC ABOUT WHAT YOU CAN DO IN THE GIVEN TIME. |
| Software and hardware requirements | 1. Specify the hardware you require to run your program on. You must include all input, processing, output and storage devices required to run this program. 2. Specify the software you require to run your program on. If your program relies on a particular version of a software, identify this. 3. List any additional utilities that would be required to run this program. |
| Success Criteria | 1. List all the things that the finished program should do if it’s successful including functionality and usability. This list will be used as part of testing the test plan table and test evidence to see whether you implemented what you planned. 2. Look over your success criteria and check that each one is measurable i.e. can you actually prove that it works? E.g. ‘There should be an on-screen help feature’ – this is provable by a print screen. However, ‘it should be easy to use’ is not provable – instead you could say ‘there should be no more than two clicks between all sections of the program to make it easy to use’. This is provable through the use of print screens/top down design of the finished product. |

|  |  |
| --- | --- |
| **Design** | |
| Decompose the problem | 1. Create a top-down design to systematically decompose the problem into smaller problems – remember each one should eventually be coded as a single sub-routine so keep this in mind as you are decomposing. |
| Structure of the solution | 1. You will need to explain each sub-program and how all of them work together. |
| Inputs, Processes, Outputs and Storage | 1. Create table that stores all the possible inputs that go into the program (keyed in, clicked, drop down list, radio button etc.) 2. In the table also list all the processing that occurs in the program (sort, search, calculate, reset/clear, retrieves, loads/opens, generates, appends, updates, saves etc.) 3. You need to also include all the outputs that occur in the program (displays, prints) 4. Finally, you need to include any data that would be stored, in which format and where. |
| Algorithms | 1. Create an algorithm for each of the sub-programs (this should be a procedure or function). If you are struggling to code it then it might be because you need to further decompose the problem.   \*\* look online for support with this section – you might have to look at similar existing codes to help you think about how it would be written.   1. Explain how all the sub-routines communicate with each other. |
| Usability features | 1. Create some screen designs of how you proposed solution should look and feel. 2. Describe and justify the what features you have used to make the use of the program straight-forward (this could be annotated on the screen designs) |
| Key variables and structures | 1. Create a table that stores the key variables required for the program and explain the purpose of each variable (name, data type, what it stores and why it is needed) 2. Explain the data structures that are needed in the program (arrays) 3. Create a validation table that describes any validation that would be required in the program and explain why this would be needed (e.g. case sensitivity, presence check, alphanumeric check etc.) |
| Test data for the development – alpha testing | 1. Testing your program at several different points in the development of the program is important. It will need to be *tested during development* and *after the completion of your program*. 2. Create a table that stores the test data you will use to test the program *during development.* 3. Ensure you have considered the test data for every success criteria. Remember there may be multiple test data for each success criteria as shown in the example below! 4. Now think about any test data you could include which could ‘break’ the program and add this to your table. You want to be able to make the program as robust as possible.  |  |  |  | | --- | --- | --- | | Test data | Relation to success criteria | Explanation | | Ms Kaur | Allow name entry | To check that it accepts letters | | 123 | Do not allow numbers for name entry | To check that it rejects numbers and asks the user to re-enter their name | | “Miss Kaur” | Do not allow punctuation for name entry | To check that it rejects punctuation and asks the user to re-enter their name | | \*blank\* | Do not allow blank for name entry | To check that the program doesn’t accept an empty field and asks the user to re-enter their name | |
| Test data for beta testing | 1. Create a table that stores the test data you will use to test the program *post development.* This might be the same as the test data used for during development. 2. Ensure you have considered the test data for every success criteria. Remember there may be multiple test data for each success criteria. 3. Now think about any test data you could include which could ‘break’ the program and add this to your table. You want to be able to make the program as robust as possible. |

|  |  |
| --- | --- |
| **Development** | |
| Iterative development | Developing a computer program is an iterative process. Each sub program (procedure or function) should be developed, tested then modified as necessary before moving on to the next one. This is an agile development process. After each successful test, the client should be involved to see the outcome of the so far developed program. They should also be involved in the consultation of what needs to be modified. |
| Design | Take out your design notes.  This should have included a description of the procedures your program will have and the order in which they should be developed. |
| Develop the 1st procedure | Using your design to help you, develop your first procedure.  You must use sensible meaningful variable names to indicate their purpose.  You must incorporate suitable validation to ensure the program works under unforeseen circumstances.  You must include commenting, explaining what every line is the code does. |
| Test the 1st procedure | Use the test plan table created in the design section to test the procedure. You must use the **test data** that has been planned in the test plan table. Add another column to the right of the test plan table called ‘actual outcome’. In this column, you will write down the outcome of the test as either ‘pass’ or ‘fail’. |
| Test Evidence | Take print screens of the outcomes of the test |
| Modify | If the code hasn’t worked then you will have to adjust the code and repeat the testing. |
| Client involvement | Once the code works as expected, write a letter/email to the client explaining what has been done and how it was tested.  Get a letter/email back from the client that provides their feedback on the program so far, including any positive feedback and areas of improvement they would like to see in the next version. |
| Modify the 1st procedure | Modify the procedure based on the feedback of the client. |
| Subsequent procedures | Repeat the steps done for the 1st procedure for the rest of the procedures until the program is complete |

|  |  |
| --- | --- |
| **Evaluation** | |
| Beta Testing | Once the program is complete you will need to conduct testing using the test plan table created for beta testing.  Your testing needs to be thorough i.e. trying to break the program is a good way to check whether you have covered all bases.   * Fill in the test plan table with an ‘actual outcome’ column stating whether the test has failed or passed. * If any test fails you will need to say why it failed, provide possible suggestions of how to remedy the problem or even better yet provide evidence of how you remedied the problem. * For each test you must show print screens of the outcomes. |
| Usability features | * Draw up a questionnaire around the effectiveness of usability of your program (use some aspects of your success criteria to help you with this). * Arrange a face to face meeting with the client. * Get them to test the program and fill in the questionnaire. * Record the client’s feedback on how easy to use your program is * Write up a list of suggestions for future developments based on the clients feedback |
| Evaluation | * List the success criteria that you originally created in the analysis section of the coursework. * For each one, state how well your program matches the requirement. * In your discussion, you might mention any changes you made to the design during the development stage. * You must explain any unmet criteria or features and comment on how these might be achieved in future development. * Mention any additional features that you could add to the finished program that weren’t planned for originally explaining the usefulness of this and the approach you would take to do this. |
| Maintenance | * Write a discussion on the future maintenance of the program and any limitations in the current version. * In your discussion, include how the program might be modified to meet any additional requirements or changing requirements. * You must also comment on the maintenance features you have included in the program. |