**A level Computer Science Project**

Your final project should contain the sections below.

Use consistent Word styles for headings and subheadings so that you can automatically create the contents list.

**Refer to the project complexity guide when choosing your project.**

**Refer to syllabus sections 2c (pages 19 and 20) and 3f (pages 24 to 30) as you work on your project.**

1. Title (page 1)
2. Contents List (starting on page 2)
3. Analysis
   1. Problem identification
      1. What are you going to do? Provide a brief summary of the problem.
      2. Identify **features** that make it solvable by computational methods
      3. Why is the **overall** problem amenable to a computational approach?
   2. Stakeholders
      1. Who is your client?
         1. Describe them and their requirements
         2. What use will they make of the system?
      2. Who is the end-user?
         1. Describe them and their requirements
         2. What use will they make of the system?
   3. Research
      1. Research the problem in depth
         1. Summarise your findings
         2. Include any sources in the bibliography
      2. Are there any existing solutions to similar problems?
         1. Identify and summarise them.
         2. What can you learn from them?
      3. Identify a suitable approach to the solution
         1. Describe a number of suitable approaches to the solution
         2. Define your proposed solution
            1. Describe the essential features including justifying your choices
            2. Explain the limitations of the system
   4. Specification of the proposed solution. Specify and justify your choice of:
      1. Hardware requirements
      2. Software requirements
      3. Success criteria
4. Design
   1. Process design
      1. Problem decomposition using (a) structure diagram(s)
         1. End your diagrams where pseudocode becomes a more appropriate way to document the process
      2. Describe these parts of the solution using algorithms (pseudocode)
      3. Explain and justify the structure of the solution and how the algorithms produce a complete solution to the problem
   2. User interface design
      1. Describe usability features to be included in the solution
      2. Justify these choices
   3. Data design
      1. File design
         1. Decide how any data will be stored in files
            1. Name, purpose, type, size, validation rules
         2. Justify the choices you have made
      2. Data storage within the program
         1. Produce a list of the main variables that will be used within your program(s)
            1. Name, purpose, type, size, validation rules
         2. Justify the choices you have made
   4. Testing design
      1. Design and justify a test strategy that may include:
         1. Module testing
         2. Testing during development
         3. Final testing of each program
         4. Final testing of the whole system
         5. User testing – usability as well as functionality
      2. Test data
         1. For each element in your test strategy
            1. Produce a test plan (see template)
            2. Ensure your test plan covers:

Typical data

Extreme data

Invalid data

Erroneous data

1. Developing the solution (methods used)
   1. You must provide evidence of prototypes and the feedback your user gave to these
   2. Your code must be modular, annotated well, make good use of white space and have meaningful identifiers
   3. You MUST provide evidence of iterative/agile development
      1. Keep backup copies of your programs at each stage of development.
         1. Each program should start with
            1. Name and purpose
            2. Date started
            3. Date most recently edited
            4. Comments including what is does and any known issues or limitations
   4. Module development and testing
      1. Keep a copy of any programs that you used to test individual modules and the testing you carried out
   5. Your final report should include a **chronological** summary of the development of your system including:
      1. Code listings
      2. Evidence of testing (may include screen capture/videos as appropriate)
      3. Comments on the next steps including any problems you have identified and how you intend to resolve them.
      4. Identify clearly when you need to make changes to your original analysis/design and justify these changes
         1. Amended pseudocode
2. Evaluation
   1. Testing
      1. Annotated testing of the final system for function and robustness – a well-planned video or videos are a good idea (bandicam.com, shareX are useful for this)
      2. Evidence of testing by end-users and their feedback on usability as well as functionality
   2. Evaluation against success criteria
      1. Copy the list of success criteria from 5diii above
      2. Comment on each item cross-referencing to the testing to justify your claims to have met each
         1. If any success criteria have not been fully met, explain how they could be addressed in further developments
   3. Description of final product
      1. Comment on the usability of your final product using annotated evidence
      2. If any usability issues have not been fully met, explain how they could be addressed in further developments
   4. Maintenance and development
      1. Discuss the maintenance of the solution
         1. Backup
         2. Error reporting
         3. upgrades
      2. Discuss potential further developments
         1. Issues within the existing system that could be improved
         2. Extensions of the system to allow for additional uses
3. Bibliography
   1. List of sources of information used in the project.