How to do your OCR Computer Science A-Level Project the easy way

(General advice only. You don't have to use the exact formats described here.)

Choosing Your Project

Your idea does not have to be unique or fully functional - marks are for documenting the process.

Check:

- ☐ Are there three existing ways that the "problem" has been solved? (e.g. spreadsheet, web app, mobile app, desktop app, 2d/3d version?)
- ☐ Can you access the hardware/software you'll need?
- ☐ Can you split the project into clear stages?
- ☐ Will it use at least one of: OOP, database/structured storage, network connectivity?
- ☐ Will it have a user interface with multiple elements (buttons, drop-downs, etc.)?
- ☐ Will it require input validation? (range, presence, lookup, etc.)
- ☐ Can you test it for scale or expected usage patterns?

Basics

- ☐ File name: 12345_6789_LastName_AB
  - 12345 = centre number
  - 6789 = candidate number
  - AB = first two letters of your first name
- ☐ Footer: page no., centre no., candidate no., name.
- ☐ Use built-in heading styles only.
- ☐ Add project title and automatic table of contents.
- ☐ Intro paragraph: clearly state what you intend to make (first person is fine).
- ☐ Avoid vague words: things, stuff, certain features.
- ☐ Be concise: due to the fact that → because.
- ☐ Use secure online storage and backups.
- ☐ Screengrabs: use a good tool (e.g. IrfanView).
- ☐ AI use: Allowed for ideas/research. Must be referenced (date, prompt, output). Work must be your own.
- ☐ Number of pages to aim for, about 50 – 100
- ☐ Remember:
  - Justify = from several options, explain which you will go with and why, and why you will not go with and why.
  - Don't faff around wrapping text around images as it wastes time and the chances are, further edits to your text will mess up your layout.

Referencing & Malpractice

Reference everything you didn't write yourself - including AI ideas, exemplar materials, and advice from others.

AI sources must include the date, prompt and output

- If you try to pass off others' work (including AI) as your own, you risk disqualification.
- Don't post your work publicly until after results day.

Analysis (Finding out exactly WHAT I will make)

It is up to you which order you put this in… do it in a way that makes sense for your project. The general flow is about researching all the features your project could have, then narrow down to the features you want to design & develop. Think of it as a funnel.

- ☐ Research: Find three or more existing **approaches** as appropriate for your project (e.g. spreadsheet, mobile app, desktop app, 2d version/3d version etc) to solving the problem
- ☐ Identify **features** from your research of each approach (bullet point list)
- ☐ Justify which *approach* you will take based on the existing approaches.
- ☐ Identify your 'typical' **stakeholder**(s). You don't need a real stakeholder.
- ☐ Describe **stakeholder**(s) characteristics e.g. age, interests, typical needs & expectations.
- ☐ For each description of a feature (NB these should be the same as already identified)…
  - Describe how stakeholder(s) make use of your solution "The stakeholder will make use of my proposed solution by…"
- ☐ Explain why your solution is suitable "My proposed solution is appropriate for the stakeholder because…" e.g.
- ☐ Here's a *possible* way of doing this for: **functionality, robustness, usability** (the final column does not have to be complete for all rows). Use **features** from your research into existing approaches. For 'robustness', identify the need for **validation rules** or checks for **data integrity** to prevent unexpected errors and crashes.

Features identified from research and own ideas	**Success Criteria and justification**	**Limitations Justification**
1. **Tournament creation:** The ability to create and name a tournament:	a. Adding a name so it can be identified b. Enter the number of teams and team names so that teams can be identified. c. Generating a valid user URL to allow data to be viewed by others so that data cannot easily be changed by hackers d. Generate an administrative URL to enable match data to be changed by others.	The URL could potentially be guessed but at this stage I do not want the complexity of adding a login system as it is of secondary importance to the key part of this project

- ☐ Limitations (limiting the scope of your project): Remember to focus on the **key functionality** of the project that you want to develop.
- ☐ Write a paragraph to explain why you **will** include the identified **features**. You could also justify the **success criteria** and limitations.
- ☐ Write a paragraph about **features** (identified from your research into existing approaches) to explain why you will **not** include them - consider also writing about the *impact* of not including them in relation to the overall project intentions.
  - If you haven't coded anything like the kind of program you want to create, it is a good idea to include research into how to code the kind of program that you want to make (or do this as you go along in the design/development process)

- ☐ **Software**: Briefly describe 2-3 options for: choice of IDE, programming language, database etc. Justify which one(s) you will use.
- ☐ **Hardware**: If applicable, write about 2-3 hardware choices. Justify which hardware you will use.

Tournament Creation: "*They will make use of my proposed solution by* naming both the tournament and teams that are participating."

"My proposed solution is appropriate to the tournament administrator because it intends to automatically store the data for convenience and utilise network connectivity to easily cater for adding and removing teams at any point. Tournament creation is an essential part of any tournament program."

Make sure that the 'analysis' section clearly identifies WHAT you will be making. If you gave it to someone else, would they be able to design a solution for what you want to make?

- **Justify features of your project that make it solvable by computational methods, e.g. algorithms, network connectivity, storage, databases, the project can be decomposed into manageable problems that can be solved computationally etc.**

Design (HOW I will make it)

You can either design, develop and test one feature at a time or do the entire design first before development. "Reverse engineering (developing before designing)" must not be evident. Design MUST come before development. Yes, you are likely to spend more time on your writeup than the coding.

- ☐ Create sketches of your UI (often easiest to do by hand)
- ☐ From your UI, annotate possible usability features (whether or not you will implement them) and describe them and how they will assist the user.
- ☐ Justify which usability features you will implement, and which you will not implement
- ☐ Decompose the problem using a structure diagram (or bullet points). Make sure this is fairly detailed.
- ☐ Create any relevant class diagrams, database design diagrams etc as required. Give a description of each attribute/field/method etc.
- ☐ Explain and justify intended data structures, key variables etc.
- ☐ Identify all inputs/data entry points to your code. Explain required validation and why it is necessary. Justify your choice of validation method(s).
- ☐ State possible test data. Justify which test data you will use. Here is a *possible* way of doing this:

Validation description	Validation rule(s)	Test data	Expected result	Reason for test data and validation
Tournament Generation: Tournament name	Presence	[blank] Demo Tournament	Suitable error message Accepted	The test data tests all possible and likely inputs. It makes sense for tournaments to have a name so they can be easily identified. There is no point in saving a tournament without a name.

- Write your algorithms, explain how they form a complete solution and link together. **Make sure that each one is linked to each feature in your analysis section.**
- Justify choice of post-development test data e.g.

Test Explanation	Justification

I will randomly generate 1,000 tournaments and check that anyone can be accessed.	To see if the project can handle a **large number of tournaments.**

Make sure that the 'design' section clearly identifies HOW your project will be made. If you gave it to someone else, would they be able to make a solution?

Development & Testing (Making it)

- ☐ For each feature/part of a feature *from your analysis/design*…
  - Code it and comment on the code for future maintenance. Emphasise where inputs/data is validated. Use sensible variable names, classes etc. Keep it 'modular'.
  - *Using code from elsewhere is OK provided it is referenced and not substantial. Following tutorials from start to end without any modification will lose your marks.*
  - Justify what was done when coding it
  - Test it - showing both code **and** the user-interface (where possible) both 'before' and 'after' any fixes including code and the output of the code. (NB don't document minor mistakes, e.g. a missing bracket, but do document logical and runtime errors. Think about how invalid inputs could break your program.)
  - For failed tests, justify any fixes made.
  - Review progress using the word **'prototype'**
  - **Try to find a way of making it better** (e.g. more efficiently, additional functionality). This shows 'iterative' development **which is *very* important!**

If you decide to deviate from your design, **do not go back and change your original design.** Instead, explain why the change was made and write about the design and development of the change within your development section.

Post-development testing & Evaluation

- ☐ Write about: functionality, robustness, usability, limitations, maintenance…
- ☐ Annotate screen grabs in relation to 'features' (from your analysis) and post-development test-data identified in the design section showing:
  - Evidence of the final product working
  - Evidence of functionality
  - Evidence of robustness
  - Evidence of usability
- ☐ Explain how tests show that each of the **success criteria** have been met/unmet/partially met (copy your success criteria from your analysis)
- ☐ Evaluate the **effectiveness** of each met/unmet/partially met **success criteria** (from your analysis). NB do not write just about what you did but the EFFECT/IMPACT it would have on the end-user… or to put it another way, *to what extent* does each part of your solution meet the needs of the user and **why** is that? It can help to think about how good (or not) your solution is in comparison to existing solutions. Write about www/ebi.
- ☐ Justify the effectiveness of met/unmet/partially met usability features
- ☐ Comment on how any unmet/partially met criteria could be addressed
- ☐ Describe how the program could deal with limitations of potential improvements/changes
- ☐ Write about the limitations of your solution (you could refer to 'limitations' already written about in the analysis) and how it could be further developed with additional features

- [ ] Write about how maintainable the project is, e.g.
    - **C**orrective - is it modular enough, well documented enough for anyone to make fixes?
    - **A**daptive - could it be adapted in light of new technology, hardware, platforms e.g. mobile, VR etc.
    - **P**erfective - Are there inefficiencies that could be improved? Perhaps in relation to speed, scalability etc.
    - Cost - what would on-going running costs look like?

Finally:

- [ ] Make sure all third-party materials are referenced and **proofread everything**.
- [ ] Create a PDF version for submission (although do submit a word-processed version too as it is easier for the marker to navigate!)

Once your project has been submitted for marking, **you cannot make any changes**. HOWEVER, you could use AI to mark your work. BUT before deciding whether or not to do this - keep in mind how the AI may use your work (including any personal information) for training data*.

Job Done 🙄

*AI Marking

Before deciding whether or not to do this - keep in mind that:

- AI might use your work (including any personal data within it) for training data.
- Copying and pasting AI generated suggestions into your own work is malpractice.
- You MUST acknowledge its use.
- Your work must be exactly that – the product of yourself, not that of AI.

AI marking is useful for feedback to find anything you've missed out.

What follows is a rubric and code to use AI to mark your project.

Rubric for marking

Analysis (out of 10 marks)

For 1–2 Marks:

- Identified some features that make the problem solvable by computational methods.
- Identified suitable stakeholders for the project and described them and some of their requirements.
- Identified some appropriate features to incorporate into their solution.
- Identified some features of the proposed computational solution.
- Identified some limitations of the proposed solution.
- Identified some requirements for the solution.
- Identified some success criteria for the proposed solution.

For 3–5 Marks:

- Described the features that make the problem solvable by computational methods.
- Identified suitable stakeholders for the project and described how they will make use of the proposed solution.
- Researched the problem, looking at existing solutions to similar problems, identifying some appropriate features to incorporate into their solution.
- Identified the essential features of the proposed computational solution.
- Identified and described some limitations of the proposed solution.
- Identified most requirements for the solution.
- Identified some measurable success criteria for the proposed solution.

For 6–8 Marks:

- Described the features that make the problem solvable by computational methods and why it is amenable to a computational approach.
- Identified suitable stakeholders for the project, described them, and explained how they will make use of the proposed solution and why it is appropriate to their needs.
- Researched the problem in depth, looking at existing solutions to similar problems, identifying and describing suitable approaches based on this research.
- Identified and described the essential features of the proposed computational solution.

- Identified and explained any limitations of the proposed solution.
- Specified the requirements for the solution, including (as appropriate) any hardware and software requirements.
- Identified measurable success criteria for the proposed solution.

For 9–10 Marks:

- Described and justified the features that make the problem solvable by computational methods, explaining why it is amenable to a computational approach.
- Identified suitable stakeholders for the project and described them, explaining how they will make use of the proposed solution and why it is appropriate to their needs.
- Researched the problem in depth, looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research.
- Identified the essential features of the proposed computational solution, explaining these choices.
- Identified and explained with justification any limitations of the proposed solution.
- Specified and justified the requirements for the solution, including (as appropriate) any hardware and software requirements.
- Identified and justified measurable success criteria for the proposed solution.

Design (out of 15 marks)

For 1–4 Marks:

- Described elements of the solution using algorithms.
- Described some usability features to be included in the solution.
- Identified the key variables/data structures/classes (as appropriate).
- Identified some test data to be used during the iterative or post-development phase.

For 5–8 Marks:

- Broken the problem down systematically into a series of smaller problems suitable for computational solutions, describing the process.
- Defined the structure of the solution to be developed.
- Described the solution fully using appropriate and accurate algorithms.
- Described the usability features to be included in the solution.
- Identified the key variables/data structures/classes (as appropriate) and any necessary validation.
- Identified the test data to be used during the iterative development of the solution.
- Identified any further data to be used in the post-development phase.

For 9–12 Marks:

- Broken the problem down systematically into a series of smaller problems, explaining the process.
- Defined in detail the structure of the solution.
- Described the solution fully using appropriate and accurate algorithms, explaining how these algorithms form a complete solution.
- Described, explaining choices made, the usability features to be included.
- Identified and justified the key variables/data structures/classes and necessary validation.
- Identified and justified the test data to be used during development.

- Identified and justified further data for post-development.

For 13–15 Marks:

- Broken the problem down systematically, explaining and justifying the process.
- Defined in detail the structure of the solution.
- Described the solution fully using appropriate and accurate algorithms, justifying how these form a complete solution.
- Described, justifying choices made, the usability features.
- Identified and justified the key variables/data structures/classes, explaining necessary validation.
- Identified and justified test data during iterative development.
- Identified and justified any further data to be used post-development.

Development (out of 15 marks)

For 1–4 Marks:

- Provided some evidence of iterative development for a coded solution.
- Solution may be linear.
- Code may be inefficient.
- Code may not be annotated appropriately.
- Variable names may be inappropriate.
- Little or no evidence of validation.
- Little evidence of review during development.

For 5–8 Marks:

- Provided evidence for most stages of the iterative development process.
- Solution has some structure.
- Code is briefly annotated.
- Some variable and structure names are appropriate.
- Some basic validation is present.
- Some evidence of review during development.

For 9–12 Marks:

- Provided evidence of each stage of the iterative development process, linking to the analysis stage.
- Provided evidence of some prototype versions.
- Solution is modular.
- Code is annotated to explain key components.
- Most variables and structures are appropriately named.
- Validation is present for most key elements.
- Development shows review at most key stages.

For 13–15 Marks:

- Provided evidence of each stage of development, linking to analysis and justifying decisions.
- Provided prototype versions for each stage.
- Solution is well-structured and modular.

- Code is annotated for future maintenance.
- All variables and structures are appropriately named.
- Validation is present for all key elements.
- Development shows review at all key stages.

Testing for Development (out of 10 marks)

For 1–2 Marks:

- Provided some evidence of testing during development.

For 3–5 Marks:

- Provided some evidence of testing during development.
- Provided evidence of failed tests and remedial actions.

For 6–8 Marks:

- Provided evidence of testing at most stages of development.
- Provided evidence of failed tests and remedial actions with some explanation.

For 9–10 Marks:

- Provided evidence of testing at each stage of development.
- Provided evidence of failed tests and remedial actions with full justification.

Testing for Evaluation (out of 5 marks)

For 1 Mark:

- Provided evidence of some post-development testing.

For 2 Marks:

- Provided evidence of final product testing for function.

For 3–4 Marks:

- Provided annotated evidence of post-development testing for function.
- Provided annotated evidence for usability testing.

For 5 Marks:

- Provided annotated evidence of post-development testing for function and robustness.
- Provided annotated evidence for usability testing.

Evaluation of Solution (out of 15 marks)

For 1–4 Marks:

- Commented on success or failure with some reference to test data.

- Information is basic, unstructured, and supported by limited evidence.

For 5–8 Marks:

- Cross-referenced some test evidence with success criteria.
- Provided evidence of usability features.
- Identified some limitations.
- Information is presented with limited structure and supported by minimal evidence.

For 9–12 Marks:

- Used test evidence to cross-reference success criteria.
- Commented on partially or unmet criteria.
- Provided usability evidence and considered limitations.
- Presented with some structure and supporting evidence.

For 13–15 Marks:

- Fully evaluated solution using test evidence.
- Justified usability feature successes/failures.
- Considered maintenance and future improvements.
- Presented in a clear, logical, and structured way with strong supporting evidence.

Marking Instructions

For each section, provide a score and justify the score.
At the end, provide a total score and overall comment.
Respond using UK English in a universal voice.
sing UK English in a universal voice.

Code

The following uses Google's Gemini which is useful for analysing large documents.

You will need to obtain a key for Gemini's API:
https://aistudio.google.com/prompts/new_chat

Code to add (Google Docs. Extensions. Apps Script)

```
function assessExternalDocumentWithGemini() {
 var documentToAssessId = "INSERT GOOGLE DOCUMENT ID"; //  document to assess
 var rubricDocId = "INSERT GOOGLE DOCUMENT ID "; //  ID of the rubric document
 var outputDocId = "INSERT GOOGLE DOCUMENT ID"; //ID of the document as to where to put the marking
 var apiKey = " INSERT GEMINI API KEY "; //  Gemini API key. You need to update this with a new key.

  // Retrieve the document text
 var documentText = getDocumentText(documentToAssessId);
```

```
 var rubricText = getDocumentText(rubricDocId);

 if (!documentText || !rubricText) {
   Logger.log("Error: Could not retrieve document or rubric.");
   return;
 }
 var endpoint = "https://generativelanguage.googleapis.com/v1beta/models/gemini-
2.0-flash:generateContent"; // you may need to update this with a different model.

// You will need to add a prompt so that the AI knows to assess the document using
the rubric.

 var payload = {
   contents: [{ parts: [{ text: documentText + "\n\nRubric:\n" + rubricText }] }]
 };

 var options = {
   method: "post",
   contentType: "application/json",
   payload: JSON.stringify(payload),
   muteHttpExceptions: false
 };

 var response = UrlFetchApp.fetch(endpoint + "?key=" + apiKey, options);
 var json = JSON.parse(response.getContentText());
 if (json && json.candidates && json.candidates.length > 0) {
   var assessment = json.candidates[0].content.parts[0].text;
   appendToDocument(outputDocId, "=== AI Assessment ===\n" + assessment);
 } else {
   Logger.log("Error: No response from AI.");
 }
}

// Function to retrieve text from a Google Docs document
function getDocumentText(docId) {
 try {
   var doc = DocumentApp.openById(docId);
   return doc.getBody().getText();
 } catch (e) {
   Logger.log("Error retrieving document text: " + e.toString());
   return null;
 }
```

```
}

// Function to append AI assessment to the document
function appendToDocument(docId, text) {
 try {
   var doc = DocumentApp.openById(docId);
   doc.getBody().appendParagraph(text);
 } catch (e) {
   Logger.log("Error appending text: " + e.toString());
 }
}
```