

# Tackling A Level Projects in Computer Science

**AQA 7517**



PG ONLINE

# Contents

## **Chapter 1 – Starting a new project** **1**

---

Objectives.....	1
Introduction.....	1
Choosing a project title.....	2
Problems and investigations.....	3
Generating project ideas.....	4
Technical skills.....	5
Projects of an A level standard.....	9
Choosing users and people for feedback.....	9
Choosing a supervisor.....	11
Being realistic.....	12
Appropriate difficulty.....	13
Languages.....	14
Project development.....	14
Choosing an Integrated Development Environment.....	15
Authentication of work.....	16

## **Chapter 2 – The report** **18**

---

Objectives.....	18
Setting up the document.....	18
Title page.....	19
Using styles.....	20
Table of contents.....	21
Headers and footers.....	22
Referencing.....	23
Backups and versions.....	24

## **Chapter 3 – Analysis** **27**

---

Objectives.....	27
The problem.....	27
Third parties.....	28
Research.....	28
Interviews.....	30
Modelling of the problem.....	33
Project objectives.....	34
Commentary.....	37

## **Chapter 4 – Documented design** **38**

---

Objectives.....	38
What is documented design?.....	38
Structure/hierarchy diagrams.....	40
System flowcharts.....	41
Data Flow Diagrams.....	42
Object-oriented design.....	43
Database design.....	50
Algorithms.....	54
Data structures and advanced techniques.....	56
File structures.....	57
HCI (Human-Computer Interaction) / Screen designs.....	58
Hardware selection.....	59
Fully articulated designs.....	60
Carrying out your design section.....	60

## **Chapter 5 – Technical solution** **62**

---

Objectives.....	62
The technical solution.....	62
Evidencing the technical solution.....	62
Completeness of solution.....	64
Techniques used.....	65

## **Chapter 6 – Testing** **69**

---

Objectives.....	69
Introduction.....	69
Sufficient testing.....	70
Iterative testing.....	70
Post-development testing.....	71
Evidencing testing.....	71

<b>Chapter 7 – Evaluation</b>	<b>76</b>
Objectives.....	76
Effective evaluation.....	76
Evaluating the solution as a whole .....	76
Evaluating how well the requirements are met .....	77
Your evaluation .....	77
Independent feedback .....	79
Evaluation of independent feedback .....	80
<b>Chapter 8 – Final checks</b>	<b>82</b>
Objectives.....	82
Checking evidence in the report.....	82
Documentation and evidence .....	83
Proof-reading and referencing.....	84
Submitting video evidence.....	84
Submitting your report .....	85
Deadline for submitting your report.....	86
<b>Index</b>	<b>87</b>
<b>Appendix</b>	<b>90</b>
Useful Microsoft® Word shortcuts and key combinations.....	90

# Chapter 1

## Starting a new project

---

### Objectives

- Choose a project title
- Choose a stakeholder
- Create a project outline
- Understand levels of technical skills
- Also, understand:
  - How to be realistic in your project scope
  - Languages that are and are not appropriate to use
  - The design methodologies that could be used
  - Different options for IDEs
  - How your work is authenticated

### Introduction

Choosing a suitable A Level project is quite a challenge. Projects contribute to 20% of your final grade and therefore choosing the right project for you is important. In particular, for AQA, you must consider how much opportunity the project you choose gives for demonstrating your technical skills.

This guide will take you through the steps to create a successful project. It will not give you the answers, but instead show you tips and tools to help keep you on track, and evidence your project efficiently. Using this guide and the specification should give you confidence in being able to produce the best project you can.

Projects are not about quantity, but quality. Exam Boards have specific **mark schemes**. You need to show you meet each of the mark scheme points. Clear and precise documentation makes it easier for both you, your teacher and the moderator to identify where you have met **mark criteria**.

#### TIP

Ensure that you fully understand the mark scheme before starting your project.

The important focus should be on programming in a high-level text-based programming language. Your project must allow you to demonstrate your programming abilities, not just design a graphically appealing website, or use a database that is built with a graphical user interface such as Microsoft® Access®.

## WARNING

Your project problem and solution must allow enough scope to access the more advanced technical techniques (see more on this in the following section). You should discuss this carefully with your teacher.

## Problems and investigations

Whilst it is up to you what you decide to do as a project, you can ask for help from your teacher when making your decision.

The first choice you have to make is whether you would like to create a solution to a problem or carry out an investigation. The vast majority of students doing AQA projects tackle problems rather than investigations.

### Problems

Most students will find a problem that exists for a group of users or one end-user. Their project then requires analysing and developing a computational solution to the problem.

#### TIP

You will need input from an end-user or other interested people when doing your project.

### Investigations

Instead of a problem, AQA allows you to undertake an investigation of an area of interest that requires a significant amount of programming.

In order to carry out an investigation, you must have a supervisor who has some understanding of the area that will be investigated.

#### TIP

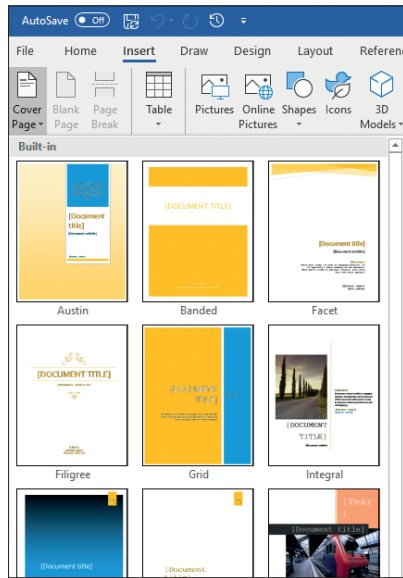
You must have a supervisor in order to carry out an investigation. Your supervisor should not be the teacher who will mark your project.

Examples of investigation projects include:

- AI or machine learning algorithms
- Data set analysis and visualisation – e.g. from live data feeds or large data sets
- Neural networks
- 3D graphics rendering

# Chapter 2

## The report



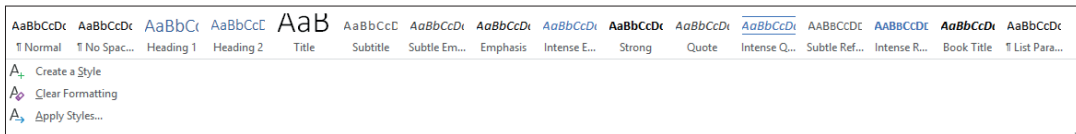
*Selection of Microsoft® Word templates*

**TASK** Create a title page for your documentation.

## Using styles

**Styles** are pre-set ways of formatting text. You should create and save any styles that are needed before adding any content to your document.

Using styles is important as they will be used later to create a table of contents for your work that will dynamically change as you add or delete pages. This will save you a lot of time.



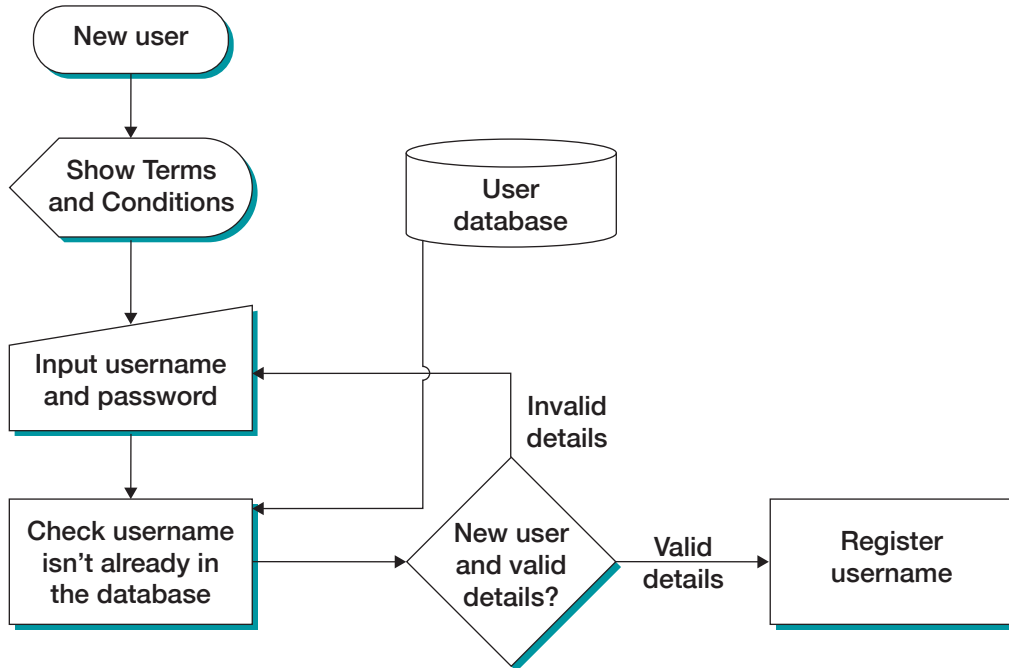
*Style options within a word processor*

The most common ones you will need are Normal, Heading 1 and Heading 2. Again, there are no extra marks for creating styles. Use the default ones that your word processor or template provide.

# Chapter 4

## Documented design

The following example shows part of a system flowchart for registration with an online booking system.



*An example of a section of a system flowchart*

### TASK

If appropriate for your project, create a system flowchart that gives an overview of how the key processes in your system work.

## Data Flow Diagrams

**Data Flow Diagrams (DFDs)** show the inputs, outputs and how data moves within your system.

Datstores are the databases or files within the system. In the analysis, the users of the system will have been identified along with the actions which they will carry out.

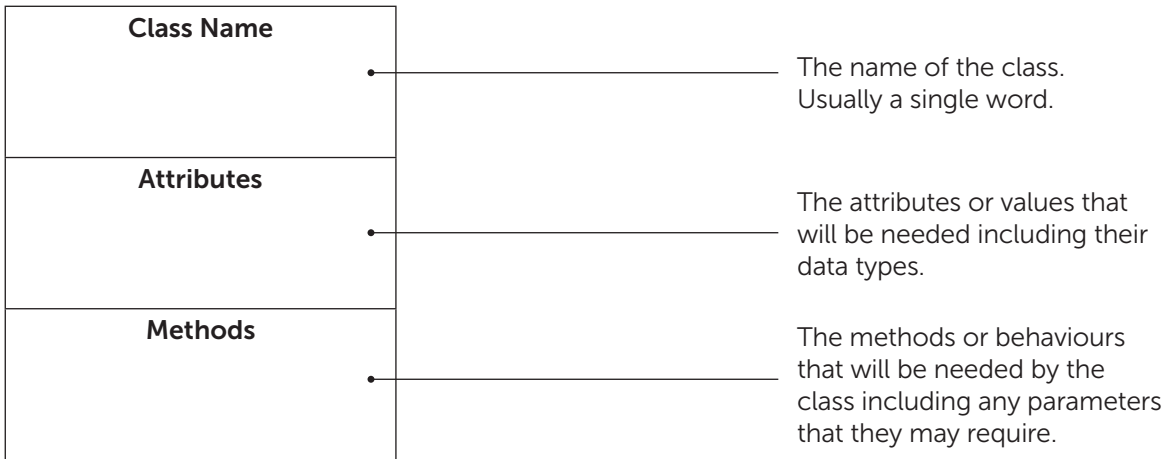
Here is an example of a very simple Data Flow Diagram for a student registering on to a course in a school:





# Chapter 4

## Documented design



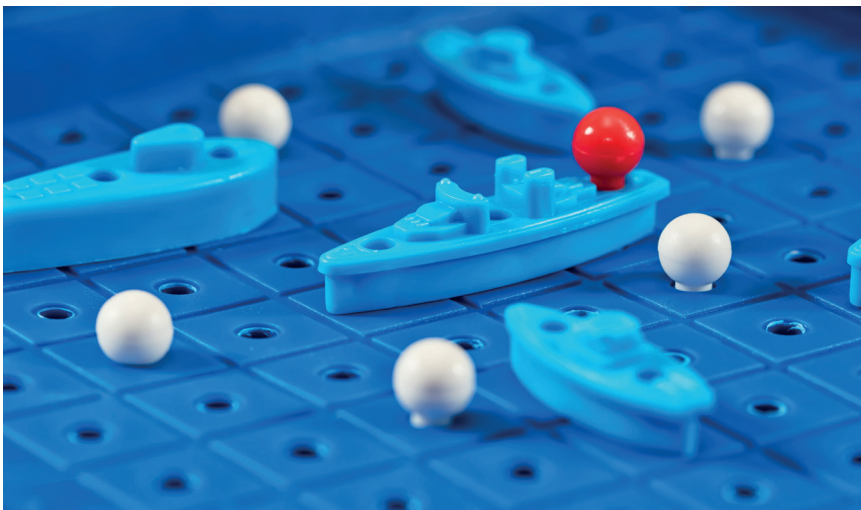
*The layout of a class*

### TIP

It is highly advisable that you have solved some simple programming problems in an OOP style before you attempt to write your own class diagrams.

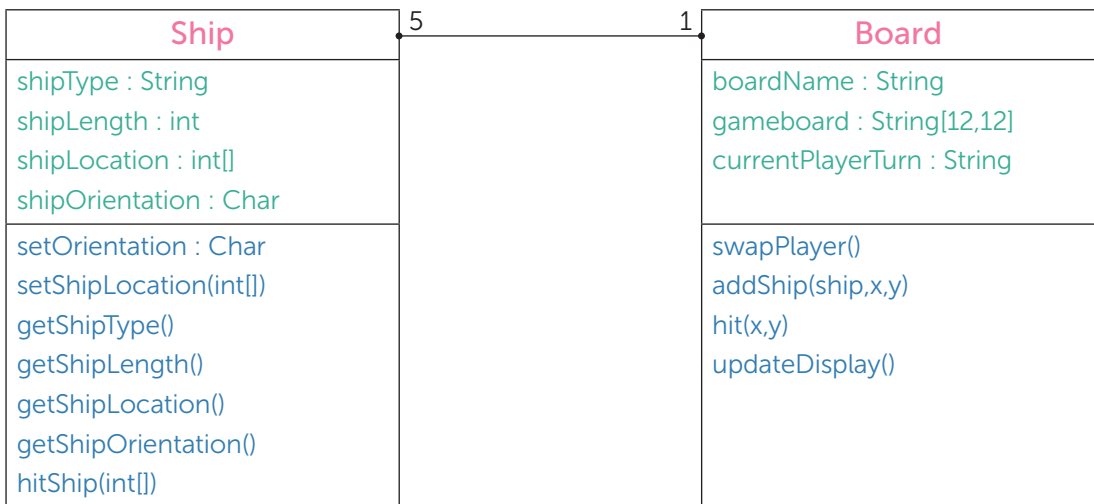
## Worked example of class diagrams

This worked example will consider a battleships style game and assume that the initial analysis of the problem has already been carried out.



## Association links and multiplicity

A `Board` class can be added which will store where ships are placed and control what happens when they are hit. 1 board has exactly 5 ships, so these numbers can be written by the **association link**. Note that in the real game, two boards would be needed, one for each player.



The number of instances that are created from each class is known as **multiplicity**.

0..1	No instances or one instance
1	Exactly one instance
0..*	Zero or more instances
1..*	One or more instances
*	Any number of instances
4..6	4 to 6 instances
3	Exactly 3 instances

### Possible multiplicity instances

Further classes would need to be built for other objects in the game such as the players.

### TASK

If your project will make use of object-oriented programming techniques then create a class diagram to describe the system.

Note that the AQA specification expects a broader understanding of OOP than that given above. If your project is heavily using OOP then when creating any designs you should consider which of the following key aspects are relevant to your project.

# Chapter 4

## Documented design

Alternatively, your solution or investigation may be saving files. For instance, if mapping data for cities were being saved with a certain file format and filename format, then you would need to explain this.

**TASK** If your project relies on more files, then show how the file structure will be organised.

## HCI (Human-Computer Interaction) / Screen designs

Designs should be given for any point where the user interacts with the computer. Typically this will be a screen design, and almost all projects require some on-screen interaction.

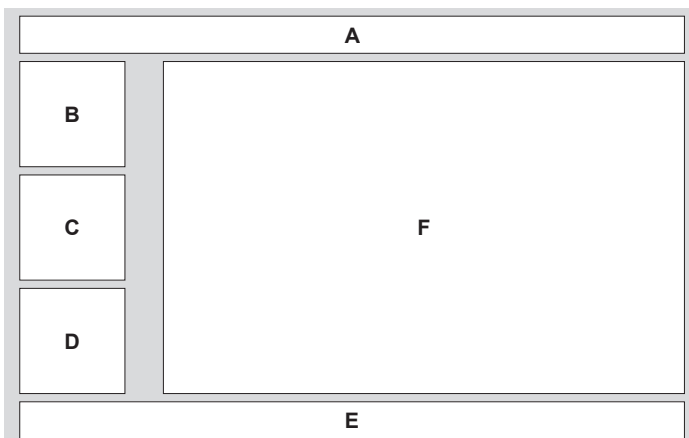
When designing a Graphical User Interface (GUI), you should consider the placement of all GUI objects. For example, in a game, this may be the menu system and placement of objects such as health bars and scores. For other systems, it may be where users enter data, view results and have menu bars placed.

The GUI is an important part of a project and should be accurately sketched out.

**TIP** You should have researched features of GUIs for similar software to that which you are designing in the Analysis section. Make sure your designs reference any applicable research that was carried out earlier.

Wireframes are a good way of designing your GUI. These are line drawings. They will help to show key features and layout clearly. Specific aspects can be numbered and explained in more detail.

If you have key themes or **house styles** that must be used in the requirements specification, then refer to these here.



- A:** Page header
- B, C, D:** Advertising
- E:** Page Footer
- F:** Page Content

### Styles

Header:

- Arial Bold 14pt, Blue

Page text:

- Arial 12pt, Dark grey

Page Header:

- Arial Bold 14pt, Black

*Sample GUI layout for web page*

# Chapter 5

## Technical solution

### Completeness of solution

You will get marks for the completeness of your solution compared to the objectives that you set out to achieve.

If some of your objectives were very hard and beyond A level standard, you will not be marked down for failing to achieve these. However, if your objectives were too easy and you meet them all, then you won't necessarily get full marks even if you do complete them all. For this reason, you should focus on having objectives that are hard enough to access technical skills from Group A.

You should aim to meet all the objectives of your system. Appropriate testing of your code will help in providing evidence that you have met the objectives.

It is a good idea to make a copy of your objectives in a table form. Create a column next to each objective which can refer to the evidence that shows that it was met including, where appropriate, a page number. The evidence may be from test results (in the next section of the report), video evidence, user feedback or code listings.

#### TIP

You can still get into the top mark band level if a few of your objectives aren't met. You need to consider whether your time is better spent fixing problems with your solution, or completing other parts of the report to a higher standard.

Objectives	Achieved	Evidence
4. Parents need to be informed of the appointments that they have just made.	✓	Testing p.63
1.1. Once they have confirmed the appointment a page will be displayed which contains 1.1.1. The appointment times 1.1.2. The names of the teachers 1.1.3. The subject for each teacher	✓	Screenshot p.65
4.2 A print button will be on the confirmation page. When clicked this will print all the appointment details given in objective 4.1	✓	Video [1m32s]

*An example showing evidence for a section of objectives being achieved*

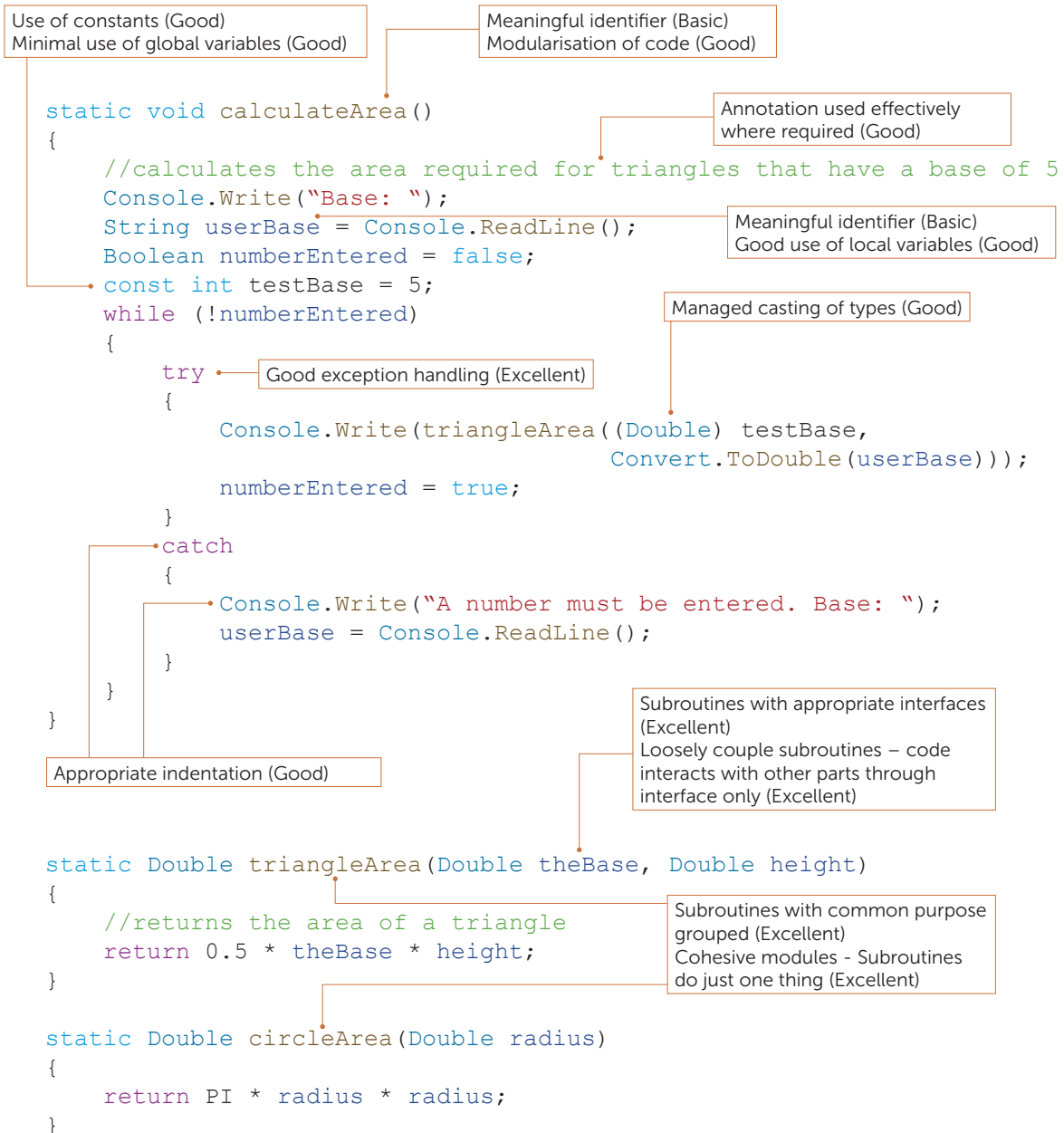
#### TASK

Create evidence to show the completeness of the solution.

# Chapter 5

## Technical solution

The following example shows a small section of code and how it can demonstrate many of the features of Basic, Good and Excellent coding styles.



# Chapter 8

## Final checks

### TIP

As your report needs to be printed, set yourself a deadline at least one working day ahead of the deadline your teacher gives you. This means that if there are technical difficulties with the printer you are using you have time to fix them.

Finally, check that each page number has printed and is in your report. If any evidence is missing then you may well lose marks. Be aware of any deadlines that your teacher has given and make sure you submit your work well before the deadline.

## Deadline for submitting your report

Deadline for printing report: \_\_\_\_\_

Deadline for handing in report: \_\_\_\_\_

### To do list

Have you done the following?

- Checked your report against the To do lists in this book
- Checked your report against the mark scheme
- Made sure that key documentation and evidence is present
- Proof-read your work
- Checked that any copied materials have been correctly referenced
- Uploaded videos
- Created a list of all video evidence
- Completed a Candidate Record Form (CRF)
- Printed the report and attached it to the CRF with a treasury tag
- Handed in the report

# Index

## A

- agile development 15
- algorithms 54
- analysis 27
- API 6, 7
- attributes 45
- authentication 16

## B

- backups 24, 25
- behaviours 45
- break down. See decomposition 40

## C

- Candidate Record Form 16
- classes 43
  - diagrams 45
  - parent 44
  - sub-classes 48
- cloud storage 25
- code
  - contents 63
  - self-documenting 67
  - styles 65
- commentary 37
- complete solution 56

## D

- database
  - Entity-Relationship (E-R) diagrams 54
  - design 50
  - normalisation 53
  - SQL 51
  - tables 53
  - validation 53
- Data Definition Language 52
- Data Flow Diagrams 42

- data structures 56
- decomposition 40
- defensive programming 67
- design 38
  - algorithm 55
  - database 50
  - designs
  - screen / HCI 58
- development
  - justification 62
- DFD. See data flow diagrams
- dialogue 9, 10
- difficulty
  - algorithms 54
- document
  - cover page 19
  - final checks 82, 90
  - font choice 21
  - headers and footers 22
  - heading 20
  - key details 18
  - page numbers 23
  - setup 18
  - styles 20
  - table of contents 21
  - title page 19
- documented design 38
- drawings 41

## E

- encapsulation 43
- Entity-Relationship diagrams 54
- evaluation 76
  - feedback 79
- exception handling 66, 67

## **F**

feedback 79  
file structures 57

## **H**

hardware 59  
Harvard referencing 23  
HCI (Human-Computer Interaction) 58  
help  
    external sources 24  
hierarchy diagrams 40  
house styles 58

## **I**

IDE 15  
    syntax highlighting 21  
inheritance 43, 44, 48  
instantiation 44  
Integrated Development Environment  
    See IDE 15  
interviews 30  
investigations 3  
iterative development 15

## **J**

JSON/XML 6, 7  
justification 62

## **L**

languages 9, 14  
    SQL 51

## **M**

mark schemes 1, 82  
meetings 31  
methods 46  
modelling 33  
modules 14

## **N**

normalisation 53

## **O**

objectives 34  
Object-Oriented design 43  
Object-Oriented Programming  
    association links 49  
    attributes 45  
    behaviours 45  
    classes 43, 44  
    inheritance 48  
    methods 46  
    multiplicity 49  
    polymorphism 43  
    sub-classes 44  
objects 44, 45

## **P**

parameterised file paths 67  
parameterised SQL 52  
photos 74  
polymorphism 43  
programming  
    difficulty 13  
problem, the 3, 27  
programming  
    language 3, 9, 14  
project 2  
    completeness 64  
    development 14  
    difficulty 5, 6  
    generating ideas 4  
    ideas 7  
    marking criteria 28  
    objectives 34  
    plan 13  
    realistic 2, 12  
    skills 9  
    suggestions 2  
    title 2  
prototyping 33



## **Q**

qualitative research 29  
quantitative research 29

## **R**

referencing 23, 84  
report  
    commentary 37  
    submitting 85  
requirements  
    exam board 14  
research 28  
    interviews 30  
    other solutions 29  
    surveys 32

## **S**

screen capture software 12  
screenshots 73  
server-side scripting 6, 7  
skills 9  
solution See problem, the  
source code 63  
SQL 51  
    aggregate 52  
    DDL 52  
    parameterised 52  
stakeholders 9  
structure diagrams 40  
styles 20  
supervisor 11, 28  
surveys 32  
syntax highlighting 21  
system flowcharts 41

## **T**

technical skills 5, 6  
technical solution 62  
testing 64  
    code 64  
    evidence 71  
    Iterative 70  
    post-development 71  
    screen capture 12  
    video 12, 72  
Third parties 28

## **U**

UML. Unified Modelling Language 45  
user 9, 28  
User Interface 58

## **V**

validation 53  
version control 24  
video 72  
video evidence 12, 84

# Appendix

## Useful shortcuts and key combinations

---

### Editing shortcut key combinations

<b>Ctrl + A</b>	Select all
<b>Ctrl + B</b>	Apply or remove bold formatting
<b>Ctrl + C</b>	Copy
<b>Ctrl + Shift + C</b>	Copy formatting
<b>Ctrl + F</b>	Find
<b>Ctrl + I</b>	Apply or remove italic formatting
<b>Ctrl + P</b>	Print
<b>Ctrl + S</b>	Save
<b>Ctrl + V</b>	Paste
<b>Ctrl + Shift + V</b>	Paste formatting
<b>Ctrl + X</b>	Cut
<b>Ctrl + Y</b>	Redo or repeat last action
<b>Ctrl + Z</b>	Undo an action
<b>Shift + F3</b>	Toggle case
<b>Alt + =</b>	Insert equation
<b>Shift + Enter</b>	Create a soft line break
<b>Ctrl + Enter</b>	Insert a page break
<b>Ctrl + [</b>	Decrease font size
<b>Ctrl + ]</b>	Increase font size

### Navigation shortcuts

<b>Ctrl + Home</b>	Go to beginning of document
<b>Ctrl + End</b>	Go to end of document
<b>Shift + F5</b>	Go to last place text was edited



PG ONLINE

Completing an A Level Computer Science project is a huge undertaking for any student regardless of their competence in programming.

The key to success is to plan and write a strong report, evidencing what has been carried out.

Tackling A Level Projects in Computer Science for AQA 7517 is the essential student guide for completing the project and, in particular, the report, with confidence and independence. It contains clear and concise instruction and examples of what needs to be included. From how to generate initial ideas and choose end users, to how to evidence your final product; this book covers it all.

This guide does not specifically teach programming and is therefore suitable for use with any language or project idea being undertaken.

With important tips and advice based on the authors' in-depth experience with Computer Science projects, this will help to keep a project's progress on track.

Finally, a guide that can help students to submit their final project with confidence before the deadline.

[www.pgonline.co.uk](http://www.pgonline.co.uk)

Schools have a **Licence to Copy**  
one chapter or 5% for teaching

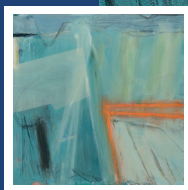


Copyright  
Licensing Agency

ISBN 9781910523209



9 781910 523209



Cover picture:

**'Long Surf Breaking'**

Acrylic and mixed media on canvas,  
100cm x 100cm © David Mankin 2018  
[www.david-mankin.com](http://www.david-mankin.com)