<TODO: All the red underlined comments are instructions. Follow these instructions one by one, starting at the start of this document and working until the end. Please delete the red underlined paragraphs as you go along answering them in turn>

Project Workbook

**Name**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Project**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Project Overview**

<TODO: Why are you doing this project?>

<TODO: What will the project build?>

<TODO: Perhaps add a nice image or something on this front page>

**Table of Contents**

[1. Definition 3](#__RefHeading___Toc106_370936444)

[1.1. Ideation 3](#__RefHeading___Toc108_370936444)

[1.2. Feasibility Study 3](#__RefHeading___Toc110_370936444)

[1.3. Requirements Specification 3](#__RefHeading___Toc112_370936444)

[1.3.1. Requirements Review Results 4](#__RefHeading___Toc162_370936444)

[2. Planning 5](#__RefHeading___Toc114_370936444)

[2.1. Project Plan 5](#__RefHeading___Toc116_370936444)

[2.2. Risk Analysis and Control 5](#__RefHeading___Toc118_370936444)

[2.3. Development Plan 5](#__RefHeading___Toc120_370936444)

[2.4. Test Plan 6](#__RefHeading___Toc164_370936444)

[3. Design 7](#__RefHeading___Toc124_370936444)

[3.1. Program Design 7](#__RefHeading___Toc126_370936444)

[3.2. Design Review 7](#__RefHeading___Toc128_370936444)

[3.3. Design Evaluation 8](#__RefHeading___Toc130_370936444)

[4. Development 9](#__RefHeading___Toc132_370936444)

[4.1. Program Development 9](#__RefHeading___Toc134_370936444)

[4.2. Code Review 9](#__RefHeading___Toc136_370936444)

[4.3. Initial Testing 9](#__RefHeading___Toc138_370936444)

[4.4. Change Records 9](#__RefHeading___Toc140_370936444)

[5. Verification 10](#__RefHeading___Toc142_370936444)

[5.1. Test Plan 10](#__RefHeading___Toc144_370936444)

[5.2. Test Results 10](#__RefHeading___Toc146_370936444)

[5.3. Test Evaluation 10](#__RefHeading___Toc148_370936444)

[5.4. User Guide 10](#__RefHeading___Toc150_370936444)

[6. Evaluation 11](#__RefHeading___Toc152_370936444)

[6.1. Project Successes 11](#__RefHeading___Toc154_370936444)

[6.2. Project Issues 11](#__RefHeading___Toc156_370936444)

[6.3. Learning Insights 11](#__RefHeading___Toc158_370936444)

[6.4. Future Work 11](#__RefHeading___Toc160_370936444)

# 1. Definition

<NOTE: The purpose of this phase is to write down exactly what it is you are building, roughly what it will do, and roughly what it will look like>

## 1.1. Ideation

<NOTE: Aligns to unit 9/B1>

Q. What is it that we are building?

Q. Who are we building it for?

Q. What will then do with it?

Q. What data will be entered before the program is used?

Q. What data will be entered when the program is being used?

Q. What sorts of outputs, screens, messages, reports will be generated?

Q. Draw a diagram of the workflow of the system when it is being used

Q. List any limitations or performance requirements

Q. Think of two different tools you could use to build this system with.

## 1.2. Feasibility Study

<NOTE: Aligns to unit 9/B2>

<TODO: Using the two different tools you identified in the previous section, build a table that answers these questions side by side for each of the two approaches. Score each proposed approach against each of these points>

Q. What technology is needed?

Q. What skills will you need to build this?

Q. What skills will users need in order to use your system?

Q. What are the health and safety implications? (e.g. RSI, trip hazards of wires)

Q. What are the data protection implications? (are you storing personal data?)

Q. How long will it take you to develop the system?

Q. What are the security requirements of the system?

Q. What are the usability implications? (e.g. do users need special training?)

<TODO: Based on the total scoring, choose the best approach, and write a short paragraph highlighting which approach you have chosen and the main reason(s) why you chose that approach over any other approach>

## 1.3. Requirements Specification

<NOTE: Aligns to Unit 9/B3>

<NOTE: It is important that you say what you are going to build, before you start to build it. A requirement is something that the customer wants the system to do for them>

Q. What is the purpose of this system?

Q. Who will use the system?

Q. What does the system do for the users?

Q. What are the key features of the system?

Q. List all inputs and outputs that you know about at this stage (either draw sketches of screens and printed repots or show sample inputs and outputs)

Q. What are the key algorithms you think you will need (e.g. sorting, finding maximum)

Q. How will data entry errors be dealt with?

Q. How will power fail of the system be dealt with?

Q. List 2 or 3 success criteria, (i.e. things that if you can say your system does these things, then the system is a success)

### 1.3.1. Requirements Review Results

<TODO: Get someone to review your requirements specification and to give you constructive feedback on it. List their feedback comments here. Constructive feedback would normally highlight some good parts of your work, as well as identifying one or two parts where you might be able to improve things.>

# 2. Planning

<NOTE: The purpose of this phase is to work out what you need to do in order to build a system that meets your requirements spec, roughly what order to do them in, and how long you think each step will take. It doesn't have to be too detailed, typically 5-10 steps maximum>

## 2.1. Project Plan

<NOTE: Aligns with Unit 9/B1>

Q. What are the key phases of the project?

Q. Estimate the effort required in each phase.

Q. Draw a GANTT chart of the phases and assign due dates of each phase.

Q. Who will accept the system as fit-for-purpose? (e.g. your customer)

Q. Agree 2 or three 'fitness for purpose' tests with your customer and list them here.

Q. What records will you keep as the project progresses? (e.g. meeting notes, time spent on each task, date that task was started and completed)

Q. Where will test records be kept (e.g. in a later section in this document)

## 2.2. Risk Analysis and Control

<NOTE: Aligns with Unit 9/B1>

Q. What are the key risks (about 3-5) of the project (e.g. what might go wrong and what problem might that cause?)

Q. Which of those risks do you want to regularly monitor throughout the project (choose 1 or 2)

Q. How often will the plan be reviewed (for a short project, a brief review every 2 days)

Q. When will plan reviews appear (put them in the plan)

Q. Who will you review the plan with (e.g. yourself, every day, with customer, every time you start a new phase).

## 2.3. Development Plan

<NOTE: Aligns with Unit 4/D1>

Q. List the phases of the development (4-5).

Q. What work will be done in each phase?

Q. What tests will be done to ensure the system works? (refer to tests in the requirements spec)

Q. Where will the test pass/fail status be recorded (e.g. in this document or in a spreadsheet)

Q. What design documentation will be created (e.g. input and output sketches, algorithm pseudo code, data storage information)

Q. How will the code be developed (e.g. iteratively, one function at a time, with testing as you go along and more formal testing when the program is written)

## 2.4. Test Plan

<NOTE: Aligns with Unit 4/C2>

Q. How will the system be tested (e.g. test as you go along and then run formal test plan tests at the end)

Q. How will the tests be recorded (e.g. in the test results in this document)

# 3. Design

<NOTE: The design of a program is the thinking and sketching that you do, before you start writing the code. The purpose of design is to think about and answer all the key questions, so that you generate a list of things that need doing for when you sit in front of the computer and start writing the code. It is important that you think about problems and record your decisions, so that coding is mostly about turning those decisions into code that the computer can execute for you>

## 3.1. Program Design

<NOTE: Aligns with: Unit 4/B2>

Q. What programming language will be used?

Q. Will there be a menu system? If so, what items are on the menu, and briefly what will each of those items do?

Q. Design all input formats (e.g. a table listing all things that need entering, whether they are strings or numbers, and the ranges of valid values)

Q. Design all output formats (e.g. draw a mock-up of what the screens look like with all the relevant parts on them, if you have printed reports, design the layout and character positions of all items of the report)

Q. Make a list of the key modules or functions you need in the system. (e.g. you could draw this as a flowchart or other diagram, with lines showing which modules or functions use which other modules or functions. It doesn't have to be completely accurate at this stage, just draw the main ones and the main important joining lines)

Q. Show where data will be stored and in what format (e.g. in a list or dictionary, or in a text file, and what is the layout of the text file)

Q. List any 3rd party libraries you think you might need (e.g. math, csvreader)

## 3.2. Design Review

<NOTE: Aligns with: Unit 4/C2>

<TODO: Get someone technical to review your design. They should provide you with short feedback that includes comments about good parts of your design, and also includes advice on parts that might need a bit of extra work or care taken when you come to code it>

Q. What were the comments of the design reviewer?

Q. List the types of tests you will do

Q. List any input range validation tests you will do for input data

Q. List any error responses or error messages

## 3.3. Design Evaluation

<NOTE: Aligns with Unit 4/C3>

Q. Make a brief list of any changes to the design that you made, as a result of the feedback from the design review.

Q. Make a list of any things you need to remember to fix when you get to writing the code (it is ok to make a note here and fix them later, rather than having to go back and re-do this section)

# 4. Development

<NOTE: The purpose of this phase is to write and further develop the program code, and basically get it working>

## 4.1. Program Development

<NOTE: Aligns with Unit 4/C1>

Q. Turn any pseudo-code you wrote earlier, into real python code. (hint, if you paste your pseudo-code in as comments, you can then write the python code below each bit, and this will comment your program as you go along). Eventually you might list your final program here if it fits, but add that at the end of the project.

Q. Run your code and fix any syntax errors. If you consistently make the same error over and over, you might make a note of that in your evaluation later.

## 4.2. Code Review

<NOTE: Aligns with Unit 4/C2>

<TODO: Get a technical person to read your code and provide feedback>

Q. What was the feedback from the technical person?

Q. What changes will you make, and what changes will you do another time?

## 4.3. Initial Testing

<NOTE: Aligns with Unit 4/C2>

Q. When the whole program is written, try a few simple tests with it and write down the tests you tried here and what you observed. (You don't have to do your whole test plan here, try to test little parts as you go along)

## 4.4. Change Records

<NOTE: Aligns with Unit 4/C3>

Q. If you made any changes as a result of the code review, briefly describe each of them here. (what was observed by the reviewer, what was the rough change you made as a result)

Q. You might want to add some more comments to parts of your program, to make it easier to read and understand by others. If you did that make a brief note here as to which sections you added more comments to.

# 5. Verification

<NOTE: The purpose of this phase is to formally and systematically test your program against all the tests that you designed for it earlier. You should not start this phase until you have a complete program that mostly seems to work>

## 5.1. Test Plan

<NOTE: Aligns with Unit 4/C2>

Q. List a table here of each test that you performed, whether it passed or failed. If it failed, make a brief note as to what it did that it should not have done.

## 5.2. Test Results

<NOTE: Aligns with Unit 4/C2>

<NOTE: This could be a separate column in your test plan, or you could number all your tests and then list the test plan above and the test results here>

## 5.3. Test Evaluation

<NOTE: Aligns with Unit 4/C2>

Q. Write a short summary here on the ability of your program to pass the tests. (If it passes all tests, say something about how your risk analysis and development planning helped that). (If it fails some tests, write a short note identifying whether these test failures are important or not to the overall operation of the system – you might be able to say for some failures that you will put a note about it in the user guide and ask the user to not do something, for example).

## 5.4. User Guide

<NOTE: Aligns with Unit 4/C3>

Q. Briefly describe what the program does (you can copy this from earlier)

Q. Explain how to start the program

Q. Explain how to enter data into the program

Q. Explain how to interpret the results that the program generates

Q. Explain how to quit the program.

Q. If there are any known problems with the program, mention them briefly here and explain how the user can avoid hitting that problem (e.g. by not entering invalid data, or not using a specific option on the menu).

# 6. Evaluation

>NOTE: The purpose of this section is to show that you can look back over your work and understand where you have made mistakes, understand things you did well, and also understand and talk about things you might do better next time you do a project like this>

## 6.1. Project Successes

Q. How well does your program meet the requirements that you originally wrote down?

Q. Ask your users about how easy the program is to use, what did they say?

Q. Comment on the quality of the code (you can extract some words from the earlier code review and put them here)

## 6.2. Project Issues

Q. Are there any known bugs in the program, or bits that are not quite finished off yet?

Q. If anything doesn't quite work yet, make some brief notes about what you would do next to investigate and fix those bugs.

Q. Make a comment about any tricky areas of Python that caused you problems as you were using it.

## 6.3. Learning Insights

Q. Say something about whether you found the structure of the plan useful or not. If it was useful, say how it made things better. If it wasn't useful, say something about what else would need to be in the plan to make it useful.

Q. Say something about whether you found the structure of the design phase useful or not. If it helped to design before you coded, say why this helped. If it didn't help, say what else you would have needed in the design to make it more useful to your coding phase.

Q. How successful was the project?

Q. How does the time you planned, match with the time you actually spent?

Q. Did you find the communications with the customer worked well or not?

Q. Give an example of where you took a specific problem and broke it down into smaller steps (e.g. splitting an algorithm into separate functions, or using a menu system)

## 6.4. Future Work

Q. If you had more time to spend on this project, list 2-3 things you would do next, and say why you would want to do these.

TODO: When you get to the end, remember to update the table of contents if it hasn't updated automatically (there is a menu option or right-click method in both Microsoft Word and LibreOffice that does that)