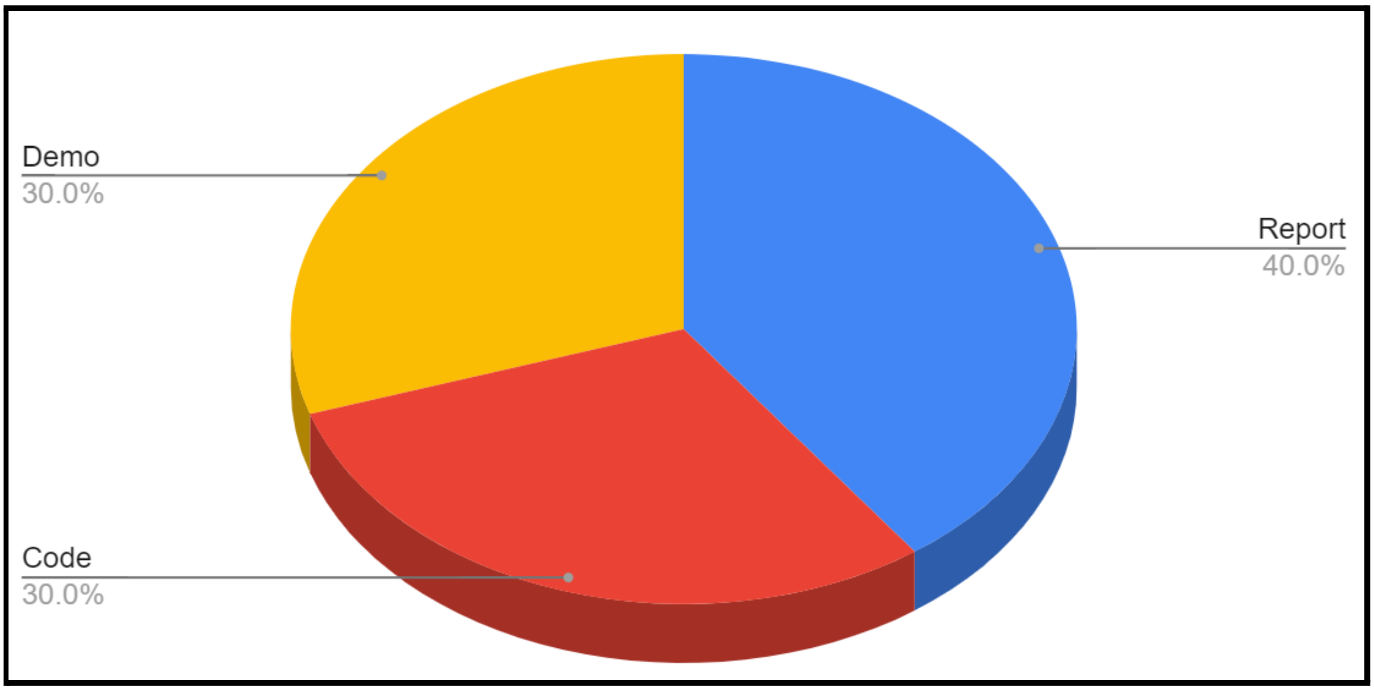
Context

Breakdown:



Brief:

Client Brief 2: Quote Calculator for flooring company
A local flooring company provides Wood, Tile & Carpet flooring. They charge as follows.

Material	Price (per m^2)	Installation cost (per $5m^2$)
Carpet	*£5*	*£50*
Wood	*£10*	*£75*
Tiled	*£15*	*£100*

Data Objects Action Context

The company wants a simple quote calculating application where they can input room dimensions and the software can provide a quote for each material. The application should be able to:

- Add a new Customer with their name, phone number & address
- List customers names with their most recent quoted price (if any)
- Update the dimensions of the floor required
- Calculate customers total cost to have the floor fitted in each different type
- Show Customer details
- Show customers historic quotes
- Show a list of all Customers in sorted by the most expensive quote to the cheapest
- Save the customer details to file
- Read the customer details from a file
- Exit Program.

Data can be stored in a file called "floorscustomer.txt"

Task 1

Once you have chosen the client brief you want to work on, produce a report. The content of which covers:

- *The design process for the code you are going to write, using suitable descriptions and diagrams, you should justify the decisions you have made, including your choice of data types etc. It is up to you to design the application in the best way you see fit. Use common sense. Write about design decisions in your report*
- *A description of the tests you will perform to evaluate the code, with explanations for the suitability.*

Then once you have completed your code:

- *Results of the tests you have performed, including any relevant screenshots etc.*
- *A brief review of your code highlighting areas for improvement*

Task 2

Write the software to fulfil the client's requirements. Use classes or structs for storing information and writing functions. Code must be written in C++. Make sure to write the code neatly and use comments as appropriate.

Task 3

You will be required to demonstrate your code, either in person in a timetabled lab session, or you could record a video of your application working with you narrating the actions being carried out. If creating a video, be sure to show all aspects of the application working.

Submission

- Submit your report to Turnitin as a Word Document.
- Submit your code as a C++ file to CodeGrade. The file should be called practicalcoursework1.cpp . Be sure to submit the C++ file containing the code and not any other visual studio file.
- If you decide to record a video submit that in the format it has been recorded in. Preferably mp4. This will be submitted through Panopto. Links will be available on blackboard.

Notes

Think about your chosen client specification carefully. Think about how it would work in the "real world".
Feel free to be a bit creative with your solutions, if you think of extra features to add then please feel free to try them. But be sure to complete everything that has been specifically requested first. Writing software is a creative process and things can change, don't be afraid to go back and change your design.
It is important to design first, as you should know what you are aiming for when you start writing your software. When writing code, you should always know what you are trying to build.
Sometimes, less is more. Don't overcomplicate things and try to reuse code as much as possible.
Read the marking RUBRIC carefully, this will show you how we will mark the assessment.

Rubric

Report 40%	☐ Very little evidence of a design process. Design is deeply flawed ☐ No Evidence of Testing ☐ No Evidence of code review ☐ Very Poor grammar and writing style. Report is very unorganized	☐ Some evidence of a design process being followed ☐ Little evidence of testing ☐ Little evidence of code review ☐ Poor grammar and writing style. Report is unorganized	☐ A suitable process has been followed. Creating a suitable design. There may be some shortcomings or the design may drift from requirements to some extent ☐ Some evidence of testing ☐ Some evidence of code review ☐ Satisfactory grammar and writing style. Report is somewhat organized	☐ A suitable design process has been followed and is clearly evident. The resulting design is clear and may show some elegance Few shortcomings. Required functionality met ☐ Good evidence of testing ☐ Good evidence of code review ☐ Good grammar and writing style. Report is organized	☐ A clear well-thought-out design for the programme is followed well. Result is clear and well presented code consideration has been made of the requirements and where ambiguity lies this has been overcome through thoughtful process and speaking to the tutor. The link between design and code is very clear with little drift ☐ Very Good evidence of testing ☐ Very Good evidence of code review ☐ Very Good grammar and writing style. Report is well organized, and coherent	☐ A clear well-thought-out design for the programme is followed well. Consideration has been made of the requirements and where ambiguity lies this has been overcome through thoughtful process and speaking to the tutor. The link between design and code is very clear with little drift. May address additional issues without impairing achievement of the required functionality or creating unnecessary complexity ☐ Excellent evidence of testing ☐ Excellent evidence of code review ☐ Excellent grammar and writing style. Report is well organized, and coherent
Code 30%	☐ Very poor. Code does not compile or run ☐ Very poor choice of data types and structures ☐ Very poor use of classes	☐ Poor. Code compiles & runs, but the software does not fulfil the clients requirements ☐ Poor choice of data types and structures ☐ Poor use of classes	☐ Satisfactory. Code compiles & runs, but the software does only partially fulfil the clients requirements ☐ Satisfactory choice of data types and structures ☐ Satisfactory use of classes	☐ Good. Code compiles & runs, but the software fulfils most of the clients requirements ☐ Good choice of data types and structures ☐ Good use of classes	☐ Very good. Code compiles & runs, but the software fulfils all the client's requirements ☐ Very good choice of data types and structures ☐ Very good use of classes	☐ Excellent. Code compiles & runs, but the software fulfils all the clients requirements very well. Software is simple to use ☐ Excellent choice of data types and structures ☐ Excellent use of classes
Demonstration/ Video 30%	☐ Very poor. Code did not compile or other functional issues ☐ Very poor voice over/ discussion	☐ Poor. Code did not compile or work correctly. However, issues small and student can see how improvements code be made ☐ Poor voice over/ discussion	☐ Satisfactory. A code compiles and operates generally to requirements. There may be some issues Student shows basic understanding of how issues could be fixed ☐ Satisfactory voice over/ discussion	☐ Good. Code compiles and operates to requirements. There may be some slight issues. Student shows good understanding of how issues could be fixed, or what improvements could be made to the code ☐ Good voice over/ discussion	☐ Very good. Code compiles and operates., very few issues. Interface should be well presented.Student can describe the code well ☐ Very good voice over/ discussion	☐ Excellent. Code compiles and operates well. Interface is easily used and operation clear. May implement additional functionality where this doesn't go beyond the spirit of the task or impair functionality required. Student can explain the code excellently ☐ Excellent voice over/ discussion