# **Borwell Software Challenge Design Documentation**

Applicant: Ishaq Olah

Introduction: This is my attempt at the software challenge for Borwell. The program designed will greet the user to tell them what the program does, it will then ask for some inputs for a rooms dimensions, including width, height and length, which will then be used by the program to calculate the amount of paint required to paint the room, as well as the volume of the room, and area of the floor. The aim with this program is to show the efficiency of my coding.

This will be designed and implemented using C#.

Problem analysis:

For this project there are some requirements as stated on the website for Borwell

* The program takes user inputs for a room’s dimensions
* The area of the floor is output
* The amount of paint required to paint walls is output
* The volume of the room is output
* Unit tests are considered
* The code is clean and readable
* The project is developed in C#, Java or C++

Assumptions:

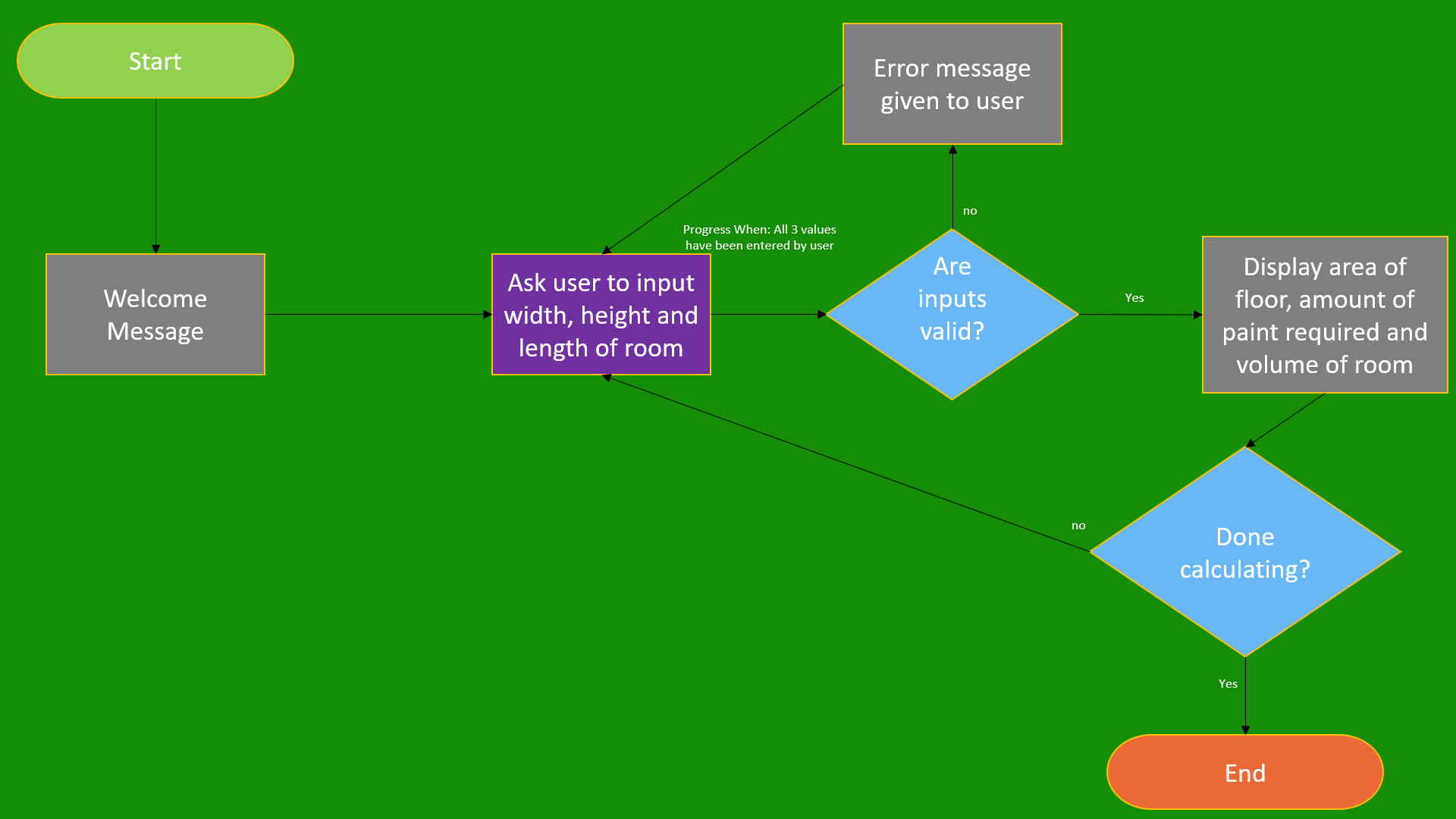
All the rooms are standard cuboid shapes without any awkward walls which could change the way volume is calculated – All walls have the same area

The amount of paint required will be assumed as 1 litre of paint for every 10m2 of wall space. It’s also going to be assumed only 1 coat is used, and only the walls are being painted, not he ceilings or floors, this standard value was derived from B&Q’s online tool (<https://www.diy.com/help-advice/wall-painting-calculator/Dev_npcart_100008.art>)

All rooms have no doors or windows which will take up space which is not going to be painted

All rooms have to have walls, and can’t just be an area of floor

Software Design



In the screenshot above, you can see a data flow diagram which has been used to plan the flow for the program based on the user’s interactions

# Testing results

Currently I am inexperienced with developing unit tests, however through some basic tests I’ve conducted myself, I’ve managed to find some bugs or potential issues

If the user inputs either just a decimal point, or multiple decimal points, the program will crash.

Some validation code was added to each of the key\_press events for the text boxes to prevent more than 1 decimal point, and code was added to the Numbers\_Only method to ensure the only value in a textbox isn’t just a decimal point

A user can currently input values which are excessively large, and unrealistic. This will also cause the numbers to be too large to be viewed entirely within the text boxes, and if the boxes were to change size, this would change the formatting of the program making it look less visually appealing. As a result, I have limited the number of characters a user can input to 5, as this should limit any issues.

After fixing these issues I realised a user can paste erroneous data into the text boxes, so I have now prevented that as well to avoid bugs.

# Overview

* The program functions as intended
* The programs functions meet the set criteria
* A few extra tasks were added in for fun, such as music and being able to convert units (which is why there are combo boxes in the program) – converting units however was not completed to be functional within the time I had spent on the application (to be updated)
* There are currently no unit tests or exception handling (also to be updated)
* I honestly forgot to start this project in GitHub (I haven’t used it frequently enough so it’s force of habit to just make a project and start normally) so there are no commits which will show the progress of each build – I’ve tried to explain the process within the testing results section of this document.