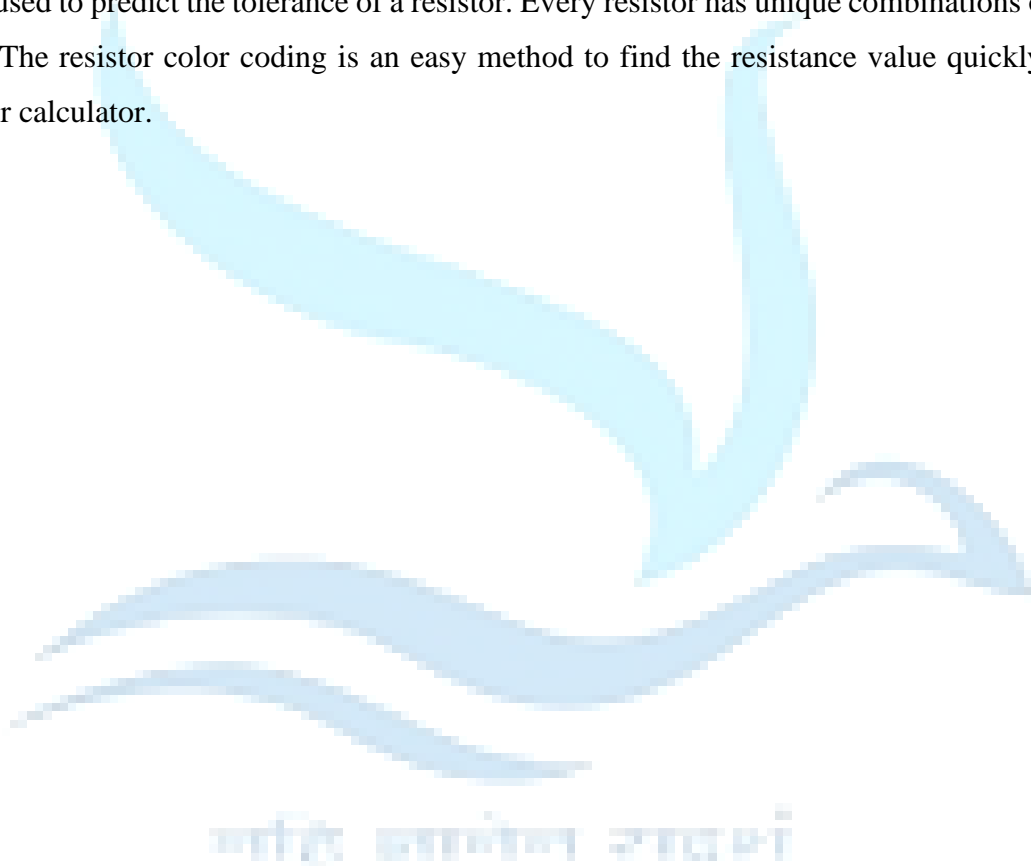


## **1.PROJECT ABSTRACT**

The resistor is an electronic component that provides electrical resistance. It also acts as a protection device in a circuit. The resistor's role includes reducing current flow, operation, adjusting signal levels, and dividing the voltage. There are many different types of Resistors available which can be used in both electrical and electronic circuits. The resistance value, tolerance, and wattage rating are generally printed onto the body of the resistor as color bands. The task is to find the resistance of 3-band, 4-band, 5-band, and 6-band resistor. The value of the resistance is not printed on any resistor. Hence, the knowledge of resistor color code is essential to determine the value of the resistance. It is also used to predict the tolerance of a resistor. Every resistor has unique combinations of the color bands. The resistor color coding is an easy method to find the resistance value quickly using the Resistor calculator.



## **2.INTRODUCTION**

We know that a resistor is a two-terminal device and is an important component in building many electronic devices. The resistor is a component that is used to limit or regulate the flow of electric current. We commonly see resistors printed with different colors. It usually contains four bands of colors. To know the value of resistors, one must know how to calculate the resistance color code. In this article, let us learn what is resistance color code, how to read resistor color code, and examples to find resistance color code.

### **WHAT IS RESISTOR COLOR CODE?**

Resistors are usually very tiny, and it is challenging to print resistance values on them. So, color bands are printed on them to represent the electrical resistance. These color bands are known as resistor color codes. The resistor color code was invented in the 1920s by the Radio Manufacturers Association (RMA). All leaded resistors with a power rating up to one watt are marked with color bands. They are given by several bands and together they specify the resistance value, the tolerance rate and sometimes the reliability or failure rates. The number of bands present in a resistor varies from three to six. The first two bands indicate the resistance value, and the third band serves as a multiplier. In this piece of article, let us discuss how to read resistor color codes, look at an example and learn a mnemonic to remember the number sequence.

### **RESISTOR COLOR TABLE**

Colour	Digit	Multiplier	Tolerance
Black	0	1	
Brown	1	10	± 1%
Red	2	100	± 2%
Orange	3	1000	
Yellow	4	10,000	
Green	5	100,000	± 0.5%
Blue	6	1,00,000	± 0.25%
Violet	7	10,000,000	± 0.1%
Grey	8		± 0.05%
White	9		
Gold		0.1	± 5%
Silver		0.01	± 10%

## COLOR CODED RESISTOR CALCULATOR

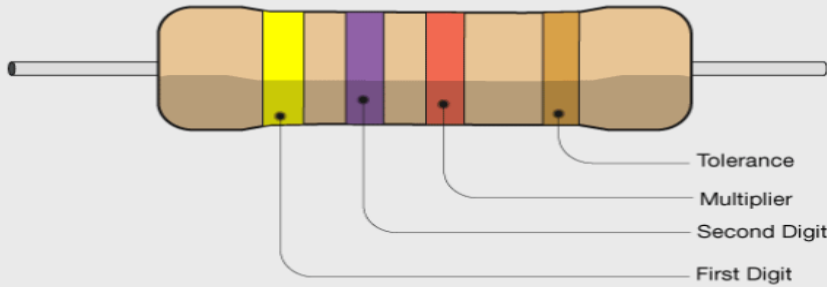
### HOW TO READ RESISTOR COLOR CODE?

To read them, hold the resistor such that the tolerance band is on your right. The tolerance band is usually gold or silver in color and is placed a little further away from the other bands.

Starting from your left, note down all the colors of the bands and write them down in sequence

Next, use the table given below to see which digits they represent

The band just next to the tolerance band is the multiplier band. So, if the color of this band is Red (representing 2), the value given is  $10^2$ .

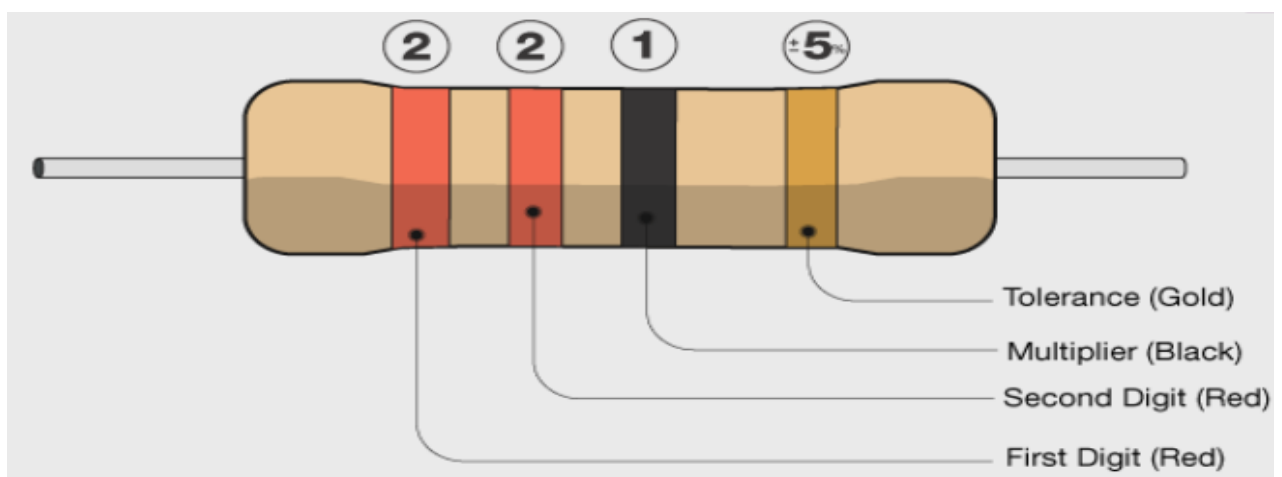
RESISTOR COLOUR CODES				
				
	1st Digit	2nd Digit	Multiplier	Tolerance
Black	0	0	$\times 1$	Silver $\pm 10\%$
Brown	1	1	$\times 10$	Gold $\pm 5\%$
Red	2	2	$\times 100$	
Orange	3	3	$\times 1000$	
Yellow	4	4	$\times 10000$	
Green	5	5	$\times 100000$	
Blue	6	6	$\times 1000000$	
Violet	7	7		
Grey	8	8		
White	9	9		

Example Shown :

Yellow	Violet	Red	Gold
4	7	$\times 100$	$\pm 5\%$

47k  $\Omega \pm 5\%$

### EXAMPLE



## COLOR CODED RESISTOR CALCULATOR

The band colors for resistor color code in the order

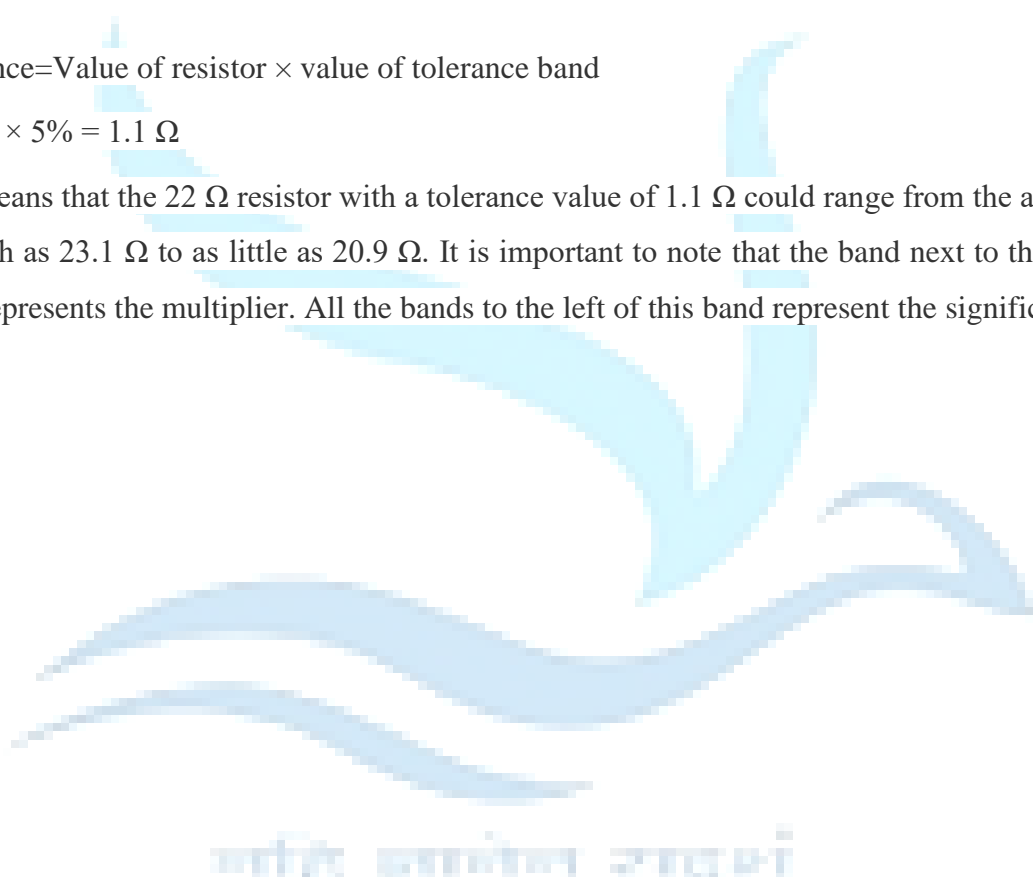
Band Colours in order	RED	RED	BLACK	GOLD
Digit representation	2	2	1	$\pm 5\%$
Value	$22\Omega \pm 5\%$			

The tolerance values represent by how much the resistance can vary from its mean value in terms of percentage. A gold band represents the lowest variation, so be sure to buy these at the electronics store. The value of the given resistance is:  $22\Omega \pm 5\%$ . The tolerance of the resistor can be calculated as;

Tolerance=Value of resistor  $\times$  value of tolerance band

$$= 22\Omega \times 5\% = 1.1\Omega$$

This means that the  $22\Omega$  resistor with a tolerance value of  $1.1\Omega$  could range from the actual value as much as  $23.1\Omega$  to as little as  $20.9\Omega$ . It is important to note that the band next to the tolerance band represents the multiplier. All the bands to the left of this band represent the significant digits.



### 3.SYSTEM REQUIREMENTS

#### 3.1. Software Requirements

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected, or unexpected from client's point of view.

##### 3.1.1. Software Requirements

- Operating System: Windows 7(min)
- IDE (Integrated Development Environment): Visual Studio 2022
- .Net Framework 4.5
- ICSharpCode.TextEditor.dll library.

#### 3.2 Hardware Requirements

- Processor – Intel i5 9<sup>th</sup> gen
- Speed - 2.8 Ghz and above
- RAM - 8 GB (min)
- Hard Disk - 40 GB
- Keyboard - Standard Windows Keyboard (QWERTY)
- Mouse - Two or Three Button Mouse
- Monitor - 1280 x 800 minimum screen resolution

## 4.SOFTWARE'S USED FOR PROJECT DEVELOPMENT

### 4.1.1. Introduction to C#

C# (pronounced "C-sharp") is an object-oriented programming language from Microsoft that aims to combine the computing power of C++ with the programming ease of Visual Basic. C# is based on C++ and contains features like those of Java. C# is designed to work with Microsoft's .Net platform. Microsoft's aim is to facilitate the exchange of information and services over the Web, and to enable developers to build highly portable applications. C# simplifies programming through its use of Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP) which allow access to a programming object or method without requiring the programmer to write additional code for each step. Because programmers can build on existing code, rather than repeatedly duplicating it, C# is expected to make it faster and less expensive to get new products and services to market.



Microsoft is collaborating with ECMA, the international standards body, to create a standard for C#. International Standards Organization (ISO) recognition for C# would encourage other companies to develop their own versions of the language. Companies that are already using C# include Apex Software, Bunka Orient, Component Source, dev Soft, FairPoint Technologies, LEAD Technologies, Proto View, and Seagate Software. C# can be written with any text editor, like Windows Notepad, and then compiled with the C# Command line compiler, csc.exe, which comes with the .NET framework.

However, most people prefer to use an IDE (Integrated Development Environment), and Microsoft offers several options for this. Their flagship is Visual Studio, which can be used to work on every possible aspect of the .NET framework. This product is very advanced and comes in several editions. Visual Studio is not exactly cheap and might even be too advanced for hobby programmers. With .NET framework 2.0, Microsoft introduced the so-called Express versions, targeted at hobby programmers and people wanting to try .NET, and they continued this tradition with the later release of .NET 3.0 and 3.5. The Express versions only work for one language, like C# or VB.NET, and miss some of the advanced features of Visual Studio. However, they are free and will work just fine for learning the languages, which is why we will use it for this tutorial. C# is a strongly typed language.

Every variable and constant have a type, as does every expression that evaluates to a value. Every method declaration specifies a name, number of parameters, and type and kind (value, reference, or output) for each input parameter and for the return value. The .NET class library defines a set of built-in numeric types and more complex types that represent a wide variety of logical constructs, such as the file system, network connections, collections and arrays of objects, and dates.

A typical C# program uses types from the class library and user-defined types that model the concepts that are specific to the program's problem domain. The information stored in a type of can include the following items:

1. The storage space that a variable of the type requires.
2. The maximum and minimum values that it can represent.
3. The members (methods, fields, events, and so on) that it contains.
4. The base type it inherits from.
5. The interface(s) it implements.
6. The kinds of operations that are permitted

In C#, memory address pointers can only be used within blocks specifically marked as unsafe, and programs with unsafe code need appropriate permissions to run. Most object access is done through safe object references, which always either point to a "live" object or have the well-defined null value; it is impossible to obtain a reference to a "dead" object (one that has been garbage collected), or to a random block of memory. An unsafe pointer can point to an instance of an 'unmanaged' value type that does not contain any references to garbage-collected objects, array, string, or a block of stack-allocated memory. Code that is not marked as unsafe can still store and manipulate pointers through the System.IntPtr type, but it cannot dereference them.

Managed memory cannot be explicitly freed; instead, it is automatically garbage collected. Garbage collection addresses the problem of memory leaks by freeing the programmer of responsibility for releasing memory that is no longer needed in most cases. Code that retains references to objects longer than is required can still experience higher memory usage than necessary, however once the final reference to an object is released the memory is available for garbage collection.

### 4.1.2. Introduction to .Net

C# programs run on .NET, a virtual execution system called the common language runtime (CLR) and a set of class libraries. The CLR is the implementation by Microsoft of the common language infrastructure (CLI), an international standard. The CLI is the basis for creating execution and development environments in which languages and libraries work together seamlessly. Source code written in C# is compiled into an intermediate language (IL) that conforms to the CLI specification. The IL code and resources, such as bitmaps and strings, are stored in an assembly, typically with an extension of .dll. An assembly contains a manifest that provides information about the assembly's types, version, and culture.

When the C# program is executed, the assembly is loaded into the CLR. The CLR performs Just-In-Time (JIT) compilation to convert the IL code to native machine instructions. The CLR provides other services related to automatic garbage collection, exception handling, and resource management. Code that's executed by the CLR is sometimes referred to as "managed code," in contrast to "unmanaged code," which is compiled into native machine language that targets a specific platform.

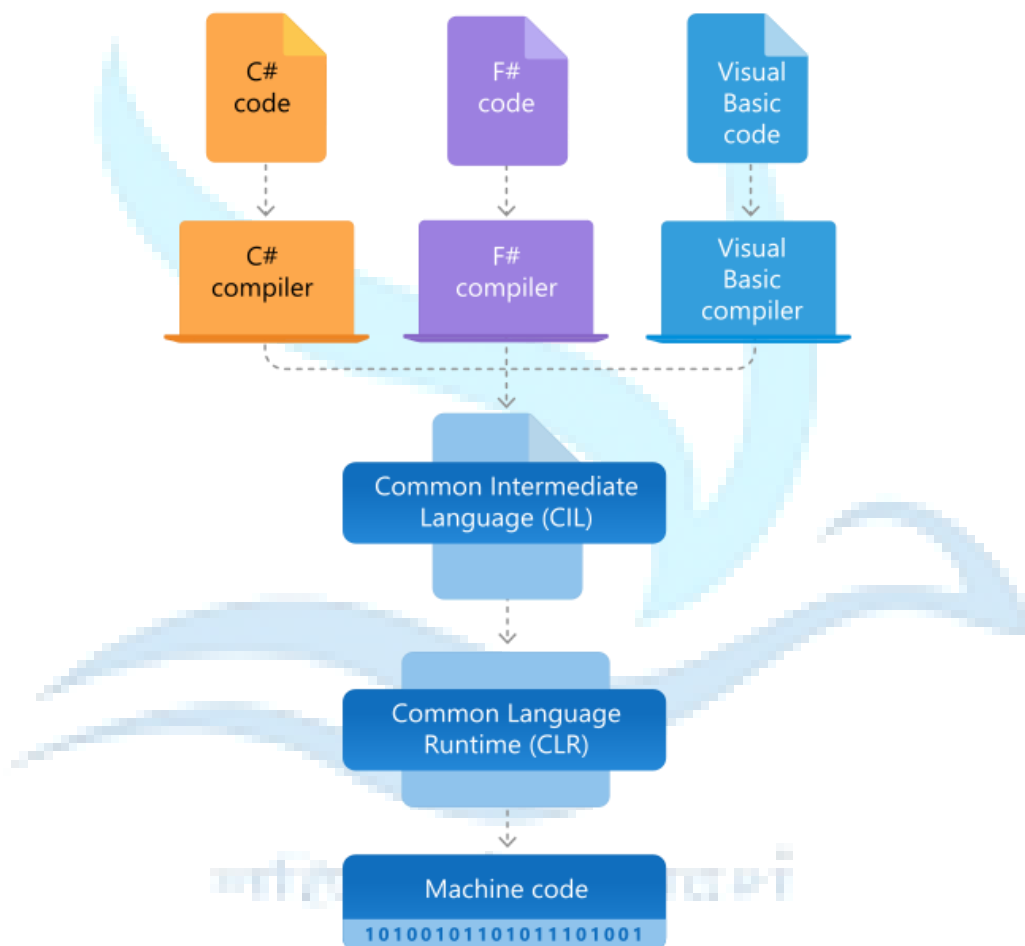
Language interoperability is a key feature of .NET. IL code produced by the C# compiler conforms to the Common Type Specification (CTS). IL code generated from C# can interact with code that was generated from the .NET versions of F#, Visual Basic, C++, or any of more than 20 other CTS-compliant languages. A single assembly may contain multiple modules written in different .NET languages, and the types can reference each other as if they were written in the same language.

In addition to the run time services, .NET also includes extensive libraries. These libraries support many different workloads. They're organized into namespaces that provide a wide variety of useful functionality for everything from file input and output string manipulation to XML parsing, to web application frameworks to Windows Forms controls. The typical C# application uses the .NET class library extensively to handle common "plumbing" chores. The two major components of .NET Framework are the Common Language Runtime and the .NET Framework Class Library.



## COLOR CODED RESISTOR CALCULATOR

- **The Common Language Runtime (CLR)** is the execution engine that handles running applications. It provides services like thread management, garbage collection, type safety, exception handling, and more.
- **The Class Library** provides a set of APIs and types for common functionality. It provides types for strings, dates, numbers, etc. The Class Library includes APIs for reading and writing files, connecting to databases, drawing, and more.



.NET applications are written in the C#, F#, or Visual Basic programming language. Code is compiled into a language-agnostic Common Intermediate Language (CIL). Compiled code is stored in assemblies—files with a .dll or .exe file extension. When an app runs, the CLR takes the assembly and uses a just-in-time compiler (JIT) to turn it into machine code that can execute on the specific architecture of the computer it is running on.

## 4.2 IDE

### 4.2.1 Microsoft Visual Studio

**Microsoft Visual Studio** is an **integrated development environment (IDE)** from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source- level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, class designer, and database schema designer.

It accepts plugins that enhance the functionality at almost every level— including adding support for source control systems (like Subversion) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer)

The logo for Microsoft Visual Studio, featuring the word "Microsoft" in a smaller font above the word "Visual Studio" in a larger font, both in white text on a purple rectangular background.


Visual Studio supports different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++ and C++/CLI (via Visual C++), VB.NET (via Visual Basic .NET), C#(via Visual C#),F#(as of Visual Studio 2010) and TypeScript(as of Visual Studio 2013 Update 2).

language services installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Java (and J#) were supported in the past. The most basic edition of Visual Studio, the Community edition, is available free of charge. The slogan for Visual Studio Community edition is "Free, fully-featured IDE for students, open-source and individual developers". As of March 2021, the current production-ready Visual Studio version was 2019, with older versions such as 2012 and 2013 on Extended Support, and 2015 and 2017 on Mainstream Support.

Visual Studio (like any other IDE) includes a code editor that supports syntax highlighting and code completion using IntelliSense for variables, functions, methods, loops, and LINQ queries. IntelliSense is supported for the included languages, as well as for XML, Cascading Style Sheets, and JavaScript when developing web sites and web applications. Autocomplete suggestions appear in a modeless list box over the code editor window, in proximity of the editing cursor. In Visual Studio 2008 onwards, it can be made temporarily semi-transparent to see the code obstructed by it. The code editor is used for all supported languages.

The Visual Studio Code Editor also supports setting bookmarks in code for quick navigation. Other navigational aids include collapsing code blocks and incremental search, in addition to normal text search and regex search. The code editor also includes a multi-item clipboard and a task list. The code editor supports code snippets, which are saved templates for repetitive code and can be inserted into code and customized for the project being worked on. A management tool for code snippets is built in as well. These tools are surfaced as floating windows which can be set to automatically hide when unused or docked to the side of the screen. The Visual Studio code editor also supports code refactoring including parameter reordering, variable and method renaming, interface extraction, and encapsulation of class members inside properties, among others.

Visual Studio features background compilation (also called incremental compilation). As code is being written, Visual Studio compiles it in the background to provide feedback about syntax and compilation errors, which are flagged with a red wavy underline. Warnings are marked with a green underline. Background compilation does not generate executable code, since it requires a different compiler than the one used to generate executable code. Background compilation was initially introduced with Microsoft Visual Basic but has now been expanded for all included languages. Visual Studio allows developers to write extensions for Visual Studio to extend its capabilities. These extensions "plug into" Visual Studio and extend its functionality. Extensions come in the form of macros, add-ins, and packages. Macros represent repeatable tasks and actions that developers can record programmatically for saving, replaying, and distributing. Macros, however, cannot implement new commands or create tool windows.

## COLOR CODED RESISTOR CALCULATOR

They are written using Visual Basic and are not compiled. Add-Ins provide access to the Visual Studio object model and can interact with the IDE tools. Add-Ins can be used to implement new functionality and can add new tool windows. Add-Ins are plugged into the IDE via COM and can be created in any COM-compliant languages. Packages are created using the Visual Studio SDK and provide the highest level of extensibility. They can create designers and other tools, as well as integrate other programming languages. The Visual Studio SDK provides unmanaged APIs as well as a managed API to accomplish these tasks. However, the managed API isn't as comprehensive as the unmanaged one. Extensions are supported in the Standard (and higher) versions of Visual Studio 2005. Express Editions do not support hosting extensions.



## 4.3 SDLC LIFE CYCLE (WATERFALL MODEL)

### 4.3.1. Water Fall Model

The waterfall model is a popular version of the systems development lifecycle model for software engineering. Often considered the classic approach to the systems development life cycle, the waterfall model describes a development method that is linear and sequential. Waterfall development has distinct goals for each phase of development. Imagine a waterfall on the cliff of a steep mountain. Once the water has flowed over the edge of the cliff and has begun its journey down the side of the mountain, it cannot turn back. It is the same with waterfall development. Once a phase of development is completed, the development proceeds to the next phase and there is no turning back. Source: Waterfall model was the first process model to be introduced and widely used in the software engineering to ensure the success of the project. The Waterfall model has got 6 phases and after the completion of each phase, the next phase will be started. The phases are Requirement, Design, Coding and Implementation, Testing, Deployment and Maintenance. This is a very old model, and the drawback is even if slight changes need to be done it has to be started from the beginning.

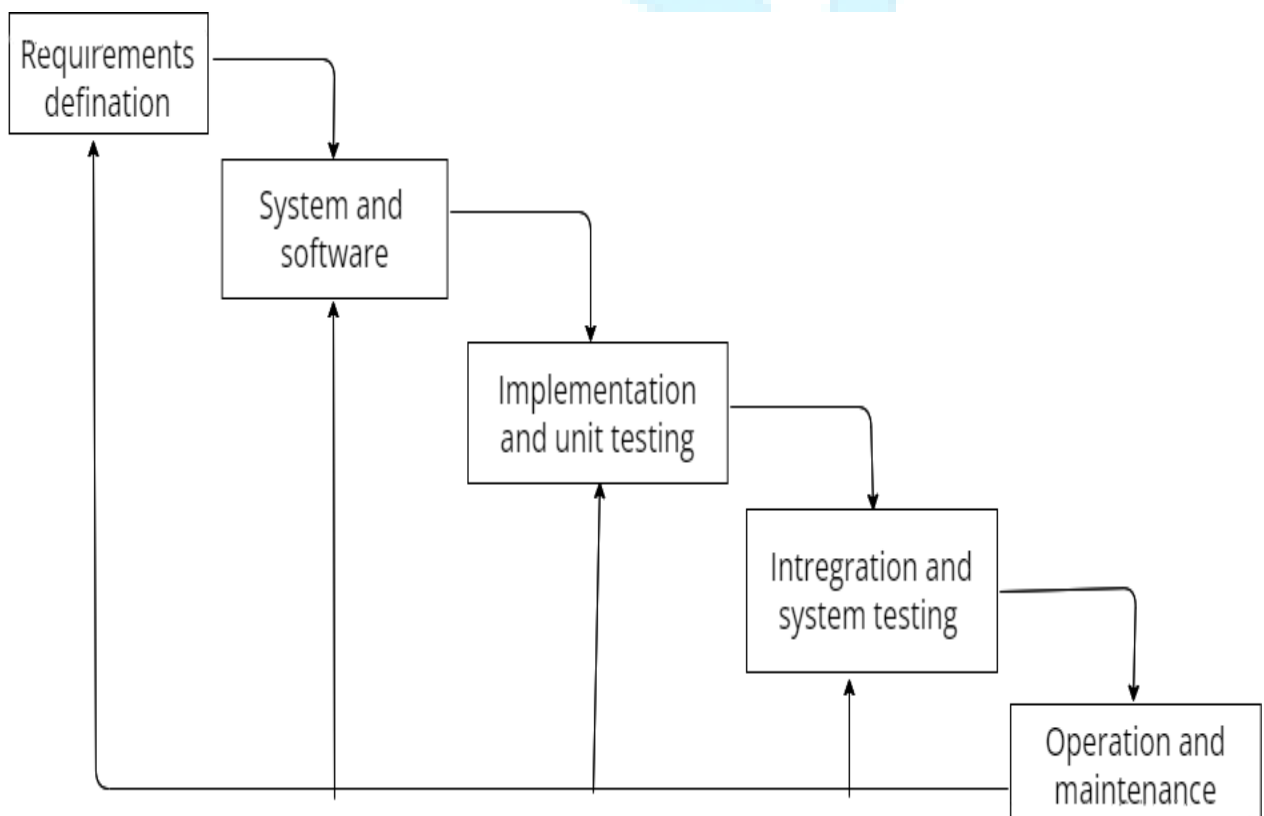


Figure 4.3.1 Waterfall model

### 4.3.2. Different Phases of Water-Fall Models

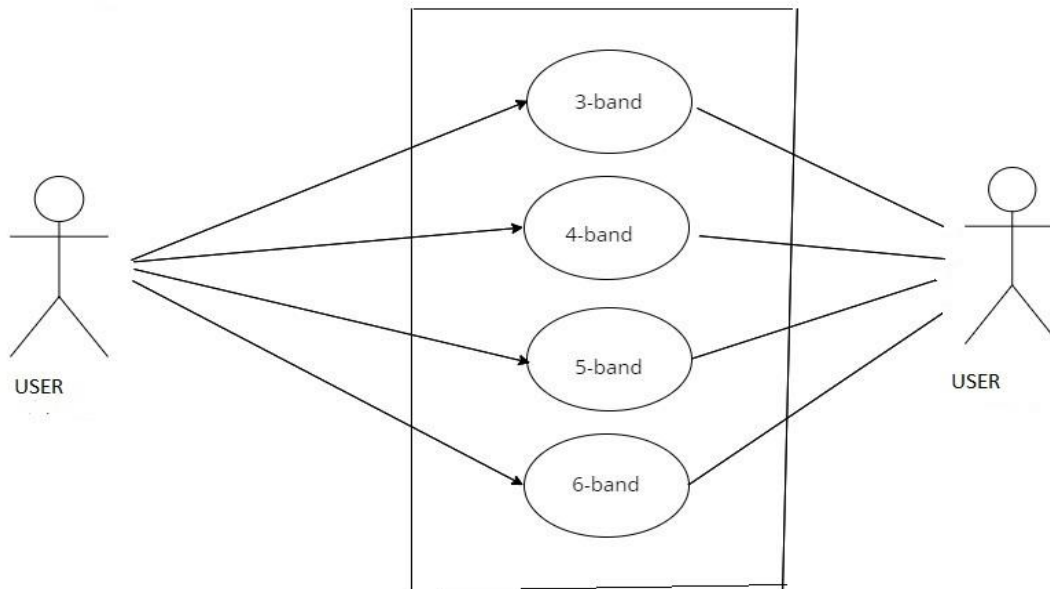
<b>Requirement</b>	This phase, detailed requirements of the software
<b>Gathering Stage</b>	System to be developed are gathered from client
<b>Design Stage</b>	Plan the programming language like Java, PHP, .net or database like Oracle MySQL, etc. Or other high-level technical details of the project
<b>Test Stage</b>	In this phase, you test the software to verify that it is built as per the specifications given by the client.
<b>Deployment Stage</b>	Deploy the application in the respective environment
<b>Maintenance Stage</b>	Once your system is ready to use, you may later require change the code as per customer request

### 4.3.3. Waterfall model can be used when

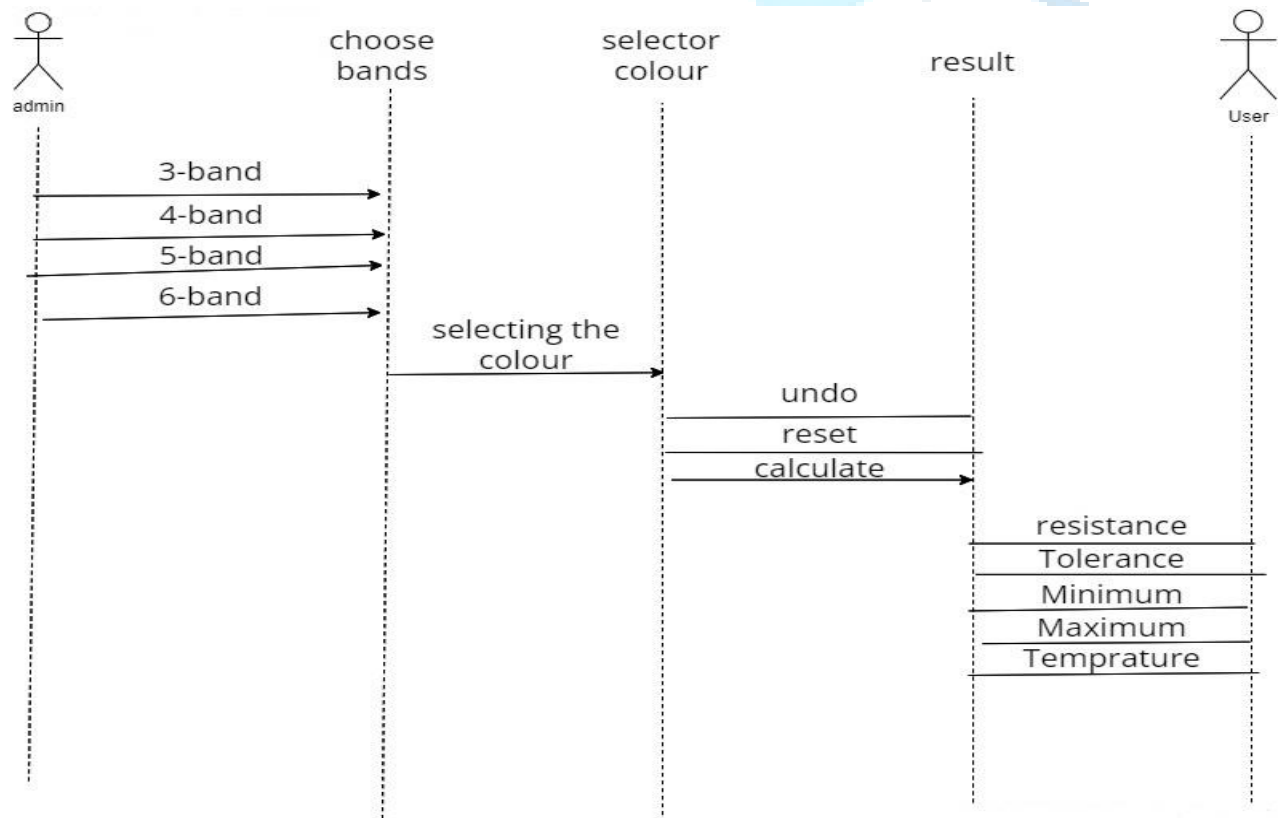
- Requirements are not changing frequently
- Application is not complicated and big
- Project is short
- Requirement is clear
- Environment is stable
- Technology and tools used are not dynamic and is stable
- Resources are available and trained

## 5. SYSTEM DESGIN

### 5.1 UECASE DIAGRAM



### 5.2 SEQUENCE DIAGRAM



## 6.SAMPLE CODING

### Login design

```
namespace ColorCodedResistorCalculator
{
    partial class login
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
        false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
```



## COLOR CODED RESISTOR CALCULATOR

```
/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()
{
    this.label1 = new System.Windows.Forms.Label();
    this.label2 = new System.Windows.Forms.Label();
    this.label3 = new System.Windows.Forms.Label();
    this.textBox1 = new System.Windows.Forms.TextBox();
    this.textBox2 = new System.Windows.Forms.TextBox();
    this.button1 = new System.Windows.Forms.Button();
    this.button2 = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // label1
    //
    this.label1.AutoSize = true;
    this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
        System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
    this.label1.Location = new System.Drawing.Point(314, 48);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(146, 25);
    this.label1.TabIndex = 0;
    this.label1.Text = "LOGIN FORM";
    //
    // label2
    //
    this.label2.AutoSize = true;
    this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.8F,
        System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
    this.label2.Location = new System.Drawing.Point(171, 184);
```

## COLOR CODED RESISTOR CALCULATOR

```
this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(119, 22);

this.label2.TabIndex = 1;

this.label2.Text = "USERNAME";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.8F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.label3.Location = new System.Drawing.Point(171, 261);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(123, 22);

this.label3.TabIndex = 2;

this.label3.Text = "PASSWORD";

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(411, 186);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(144, 22);

this.textBox1.TabIndex = 3;

this.textBox1.TextChanged += new System.EventHandler(this.textBox1_TextChanged);

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(411, 261);

this.textBox2.Name = "textBox2";

this.textBox2.Size = new System.Drawing.Size(144, 22);

this.textBox2.TabIndex = 4;
```

## COLOR CODED RESISTOR CALCULATOR

```
//  
  
// button1  
  
//  
  
this.button1.Location = new System.Drawing.Point(175, 361);  
this.button1.Name = "button1";  
this.button1.Size = new System.Drawing.Size(137, 39);  
this.button1.TabIndex = 5;  
this.button1.Text = "LOGIN";  
this.button1.UseVisualStyleBackColor = true;  
this.button1.Click += new System.EventHandler(this.button1_Click);  
  
//  
  
// button2  
  
//  
  
this.button2.Location = new System.Drawing.Point(411, 361);  
this.button2.Name = "button2";  
this.button2.Size = new System.Drawing.Size(132, 39);  
this.button2.TabIndex = 6;  
this.button2.Text = "RESET";  
this.button2.UseVisualStyleBackColor = true;  
this.button2.Click += new System.EventHandler(this.button2_Click);  
  
//  
  
// login  
  
//  
  
this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);  
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  
this.ClientSize = new System.Drawing.Size(800, 450);  
this.Controls.Add(this.button2);  
this.Controls.Add(this.button1);  
this.Controls.Add(this.textBox2);  
this.Controls.Add(this.textBox1);
```

```
this.Controls.Add(this.label3);  
this.Controls.Add(this.label2);  
this.Controls.Add(this.label1);  
this.Name = "login";  
this.Text = "login";  
this.ResumeLayout(false);  
this.PerformLayout();
```

### Calculate

```
using System;  
using System.Windows.Forms;  
using ColorCodedResistorCalculator.DataModel;  
  
namespace ColorCodedResistorCalculator  
{  
    public class Calculate  
    {  
        protected Button calculate;  
        protected char band;  
        protected const char three_band = '3', four_band = '4', five_band = '5', six_band = '6';  
  
        public Calculate(Button calculate)  
        {  
            this.calculate = calculate;  
            calculate.Enabled = false;  
        }  
  
        //Identifies the color-code value of the resistor then calculates  
        public Result CalculateBand(Panel[] resistor_colors)  
        {  
            Result result = new Result(-1, -1, -1, -1, -1);
```

## COLOR CODED RESISTOR CALCULATOR

```
Panel first_digit_color = resistor_colors[0], second_digit_color = resistor_colors[1],
third_digit_color, multiplier_color, tolerance_color, temp_coefficient_color;

switch (band)
{
case three_band:
multiplier_color = resistor_colors[2];
result = CalculateSelectedBand(first_digit_color, second_digit_color, multiplier_color);
break;
case four_band:
multiplier_color = resistor_colors[2];
tolerance_color = resistor_colors[3];
result = CalculateSelectedBand(first_digit_color, second_digit_color, multiplier_color,
tolerance_color);
break;
case five_band:
third_digit_color = resistor_colors[2];
multiplier_color = resistor_colors[3];
tolerance_color = resistor_colors[4];
result = CalculateSelectedBand(first_digit_color, second_digit_color, third_digit_color,
multiplier_color, tolerance_color);
break;
case six_band:
third_digit_color = resistor_colors[2];
multiplier_color = resistor_colors[3];
tolerance_color = resistor_colors[4];
temp_coefficient_color = resistor_colors[5];
result = CalculateSelectedBand(first_digit_color, second_digit_color, third_digit_color,
multiplier_color, tolerance_color, temp_coefficient_color);
break;
```

## COLOR CODED RESISTOR CALCULATOR

```
}  
  
return result;  
  
}  
  
#region Calculates the resistor's different bands  
  
private Result CalculateSelectedBand(Panel c1, Panel c2, Panel c3) //CalculateBand (for 3-band  
resistor)  
{  
    string concat_digits = string.Concat(GetColorValue(c1).Digit, GetColorValue(c2).Digit);  
    int digits = Convert.ToInt32(concat_digits);  
    double resistance = digits * GetColorValue(c3).Multiplier;  
  
    double min_resistance = GetMinMaxResistance(resistance).Min;  
    double max_resistance = GetMinMaxResistance(resistance).Max;  
    double tolerance = 20; // The default tolerance value of 3-band resistor is 20%  
    Result result = new Result(resistance, tolerance, min_resistance, max_resistance, -1);  
    return result;  
}  
  
private Result CalculateSelectedBand(Panel c1, Panel c2, Panel c3, Panel c4) //CalculateBand (for  
4-band resistor)  
{  
    string concat_digits = string.Concat(GetColorValue(c1).Digit, GetColorValue(c2).Digit);  
    int digits = Convert.ToInt32(concat_digits);  
    double resistance = digits * GetColorValue(c3).Multiplier;  
  
    double min_resistance = GetMinMaxResistance(c4, resistance).Min;  
    double max_resistance = GetMinMaxResistance(c4, resistance).Max;  
    double tolerance = GetColorValue(c4).Tolerance;  
    Result result = new Result(resistance, tolerance, min_resistance, max_resistance, -1);  
    return result;  
}
```

```
}

private Result CalculateSelectedBand(Panel c1, Panel c2, Panel c3, Panel c4, Panel c5)
//CalculateBand (for 5-band resistor)
{
    string concat_digits = string.Concat(GetColorValue(c1).Digit, GetColorValue(c2).Digit,
    GetColorValue(c3).Digit);
    int digits = Convert.ToInt32(concat_digits);
    double resistance = digits * GetColorValue(c4).Multiplier;

    double min_resistance = GetMinMaxResistance(c5, resistance).Min;
    double max_resistance = GetMinMaxResistance(c5, resistance).Max;
    double tolerance = GetColorValue(c5).Tolerance;
    Result result = new Result(resistance, tolerance, min_resistance, max_resistance, -1);
    return result;
}

private Result CalculateSelectedBand(Panel c1, Panel c2, Panel c3, Panel c4, Panel c5, Panel c6)
//CalculateBand (for 6-band resistor)
{
    string concat_digits = string.Concat(GetColorValue(c1).Digit, GetColorValue(c2).Digit,
    GetColorValue(c3).Digit);
    int digits = Convert.ToInt32(concat_digits);
    double resistance = digits * GetColorValue(c4).Multiplier;

    double min_resistance = GetMinMaxResistance(c5, resistance).Min;
    double max_resistance = GetMinMaxResistance(c5, resistance).Max;
    double tolerance = GetColorValue(c5).Tolerance;

    int temp_coefficient = GetColorValue(c6).Temp_Coefficient;
```

## COLOR CODED RESISTOR CALCULATOR

```
Result result = new Result(resistance, tolerance, min_resistance, max_resistance,
temp_coefficient);
return result;
}
```

```
private MinMaxResistance GetMinMaxResistance(Panel tolerance_color_panel, double
resistance)
```

```
{
double tolerance = GetColorValue(tolerance_color_panel).Tolerance;
// Formula: tolerance = (tolerance value / 100) x resistance
tolerance /= 100;
tolerance *= resistance;
double minResistance = resistance - tolerance;
double maxResistance = resistance + tolerance;
```

```
MinMaxResistance minMaxResistance = new MinMaxResistance(minResistance,
maxResistance);
return minMaxResistance;
}
```

```
private MinMaxResistance GetMinMaxResistance(double resistance) // For 3 band only that has a
default tolerance value of 20%
```

```
{
double tolerance = 20; // default tolerance value of 3 band resistor

tolerance /= 100;
tolerance *= resistance;

double minResistance = resistance - tolerance;
```



## COLOR CODED RESISTOR CALCULATOR

```
double maxResistance = resistance + tolerance;

MinMaxResistance result = new MinMaxResistance(minResistance, maxResistance);

return result;

}

#endregion

#region Assigning the values in every resistor's color

private ColorValue GetColorValue(Panel color)
{
    ColorValue black = new ColorValue(0, 1, -1, 250);
    ColorValue brown = new ColorValue(1, 10, 1, 100);
    ColorValue red = new ColorValue(2, 100, 2, 50);
    ColorValue orange = new ColorValue(3, 1000, 3, 15);
    ColorValue yellow = new ColorValue(4, 10_000, 4, 25);
    ColorValue green = new ColorValue(5, 100_000, 0.5, 20);
    ColorValue blue = new ColorValue(6, 1000_000, 0.25, 10);
    ColorValue violet = new ColorValue(7, 10_000_000, 0.10, 5);
    ColorValue gray = new ColorValue(8, 100_000_000, 0.05, 1);
    ColorValue white = new ColorValue(9, 1000_000_000, -1, -1);
    ColorValue gold = new ColorValue(-1, 0.1, 5, -1);
    ColorValue silver = new ColorValue(-1, 0.01, 10, -1);

    ColorValue colorValues;

    switch (color.BackColor.Name)
    {
        case "Black": colorValues = black; break;
        case "Brown": colorValues = brown; break;
        case "Red": colorValues = red; break;
        case "Orange": colorValues = orange; break;
        case "Yellow": colorValues = yellow; break;
```

## COLOR CODED RESISTOR CALCULATOR

```
case "Green": colorValues = green; break;
case "Blue": colorValues = blue; break;
case "Violet": colorValues = violet; break;
case "Gray": colorValues = gray; break;
case "White": colorValues = white; break;
case "Gold": colorValues = gold; break;
case "Silver": colorValues = silver; break;
default: colorValues = new ColorValue(-1, -1, -1, -1); break;
}
return colorValues;
}
#endregion
}
}
```



જાતિ કાળેલ સત્ય

## 7. OUTPUT SCREENS

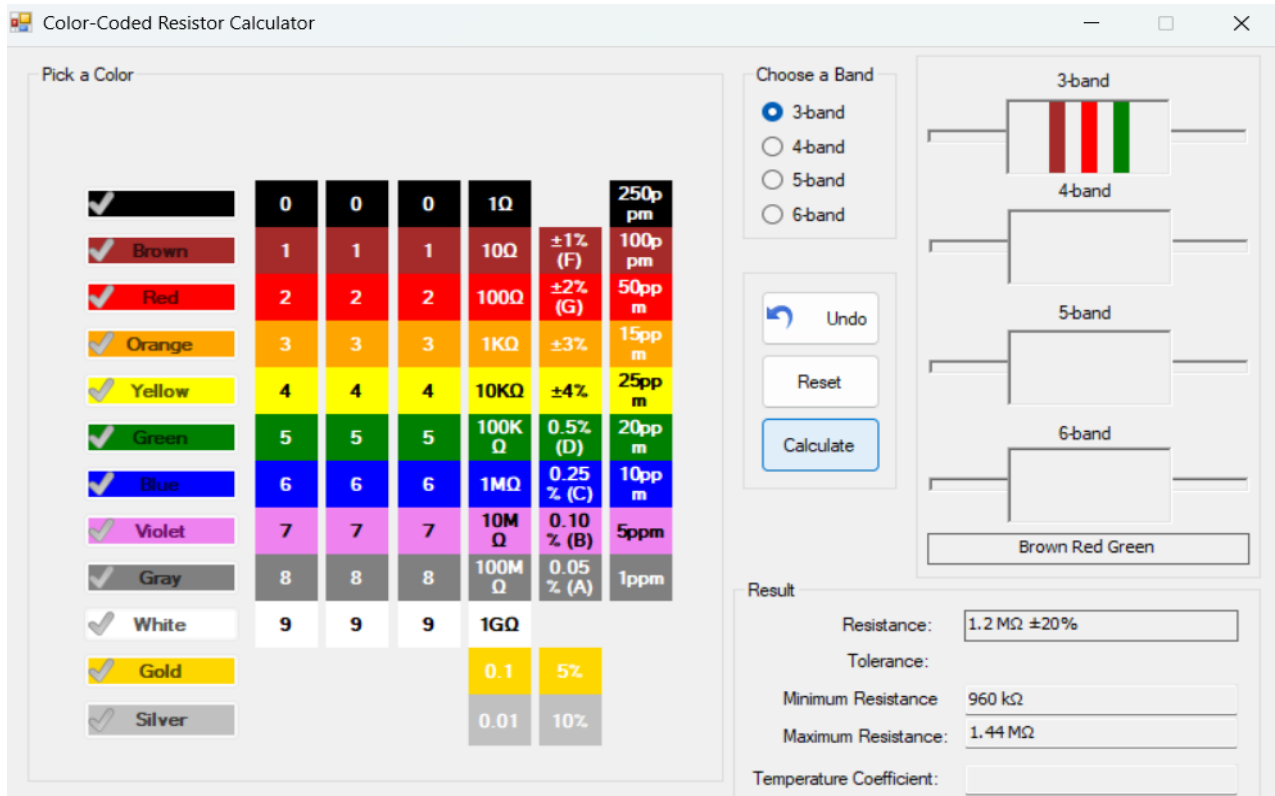


Figure 7.1 3-Band output

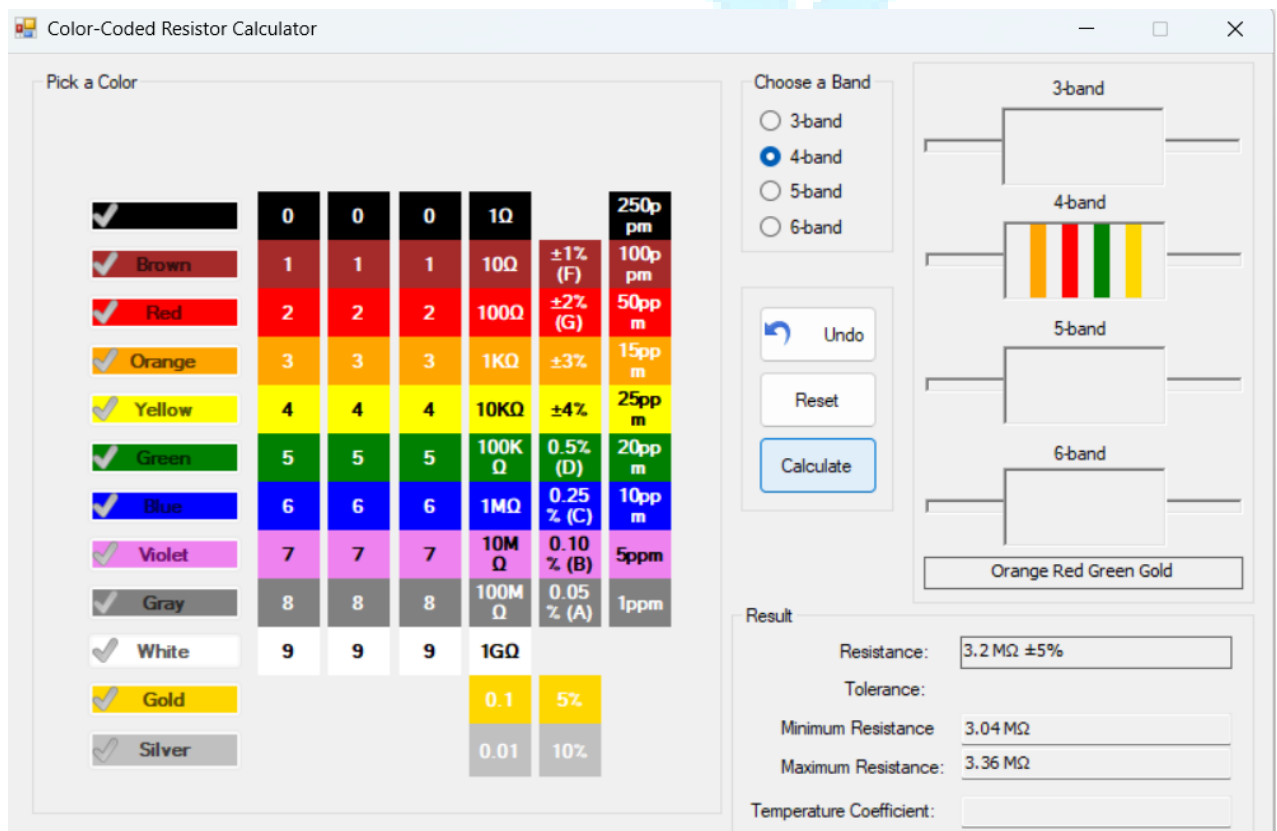


Figure 7.2 4-Band output

## COLOR CODED RESISTOR CALCULATOR

Color-Coded Resistor Calculator

Pick a Color

<input checked="" type="checkbox"/> Black	0	0	0	1Ω		250p pm
<input checked="" type="checkbox"/> Brown	1	1	1	10Ω	±1% (F)	100p pm
<input checked="" type="checkbox"/> Red	2	2	2	100Ω	±2% (G)	50pp m
<input checked="" type="checkbox"/> Orange	3	3	3	1KΩ	±3%	15pp m
<input checked="" type="checkbox"/> Yellow	4	4	4	10KΩ	±4%	25pp m
<input checked="" type="checkbox"/> Green	5	5	5	100K Ω	0.5% (D)	20pp m
<input checked="" type="checkbox"/> Blue	6	6	6	1MΩ	0.25 % (C)	10pp m
<input checked="" type="checkbox"/> Violet	7	7	7	10M Ω	0.10 % (B)	5ppm
<input checked="" type="checkbox"/> Gray	8	8	8	100M Ω	0.05 % (A)	1ppm
<input checked="" type="checkbox"/> White	9	9	9	1GΩ		
<input checked="" type="checkbox"/> Gold				0.1	5%	
<input checked="" type="checkbox"/> Silver				0.01	10%	

Choose a Band

☐ 3-band  
☐ 4-band  
☒ 5-band  
☐ 6-band

3-band

4-band

5-band

6-band

Green Yellow Blue Brown Red

Result

Resistance: 5.46 kΩ ±2%

Tolerance:

Minimum Resistance: 5.351 kΩ

Maximum Resistance: 5.569 kΩ

Temperature Coefficient:

**Figure 7.3 5-Band output**

Color-Coded Resistor Calculator

Pick a Color

<input checked="" type="checkbox"/> Black	0	0	0	1Ω		250p pm
<input checked="" type="checkbox"/> Brown	1	1	1	10Ω	±1% (F)	100p pm
<input checked="" type="checkbox"/> Red	2	2	2	100Ω	±2% (G)	50pp m
<input checked="" type="checkbox"/> Orange	3	3	3	1KΩ	±3%	15pp m
<input checked="" type="checkbox"/> Yellow	4	4	4	10KΩ	±4%	25pp m
<input checked="" type="checkbox"/> Green	5	5	5	100K Ω	0.5% (D)	20pp m
<input checked="" type="checkbox"/> Blue	6	6	6	1MΩ	0.25 % (C)	10pp m
<input checked="" type="checkbox"/> Violet	7	7	7	10M Ω	0.10 % (B)	5ppm
<input checked="" type="checkbox"/> Gray	8	8	8	100M Ω	0.05 % (A)	1ppm
<input checked="" type="checkbox"/> White	9	9	9	1GΩ		
<input checked="" type="checkbox"/> Gold				0.1	5%	
<input checked="" type="checkbox"/> Silver				0.01	10%	

Choose a Band

☐ 3-band  
☐ 4-band  
☐ 5-band  
☒ 6-band

3-band

4-band

5-band

6-band

Yellow Brown White Violet Orange Brown

Result

Resistance: 4.19 GΩ ±3%

Tolerance:

Minimum Resistance: 4.064 GΩ

Maximum Resistance: 4.316 GΩ

Temperature Coefficient: 100 ppm/°C

**Figure 7.3 6-Band output**

## 8.TESTING

### INTRODUCTION TO TESTING

System Testing involves two kinds of testing integration testing and acceptance testing. Developing a strategy for integrating the components of a software system into a functioning whole requires careful planning so that modules are available for integration when needed. Acceptance testing involves planning and execution of various tests to demonstrate that the implemented system satisfies the requirements document.

#### 8.1 Software testing Software

Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding, Testing presents an interesting anomaly for the software engineer.

##### 8.1.1. Testing Objectives include

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a probability of finding a yet undiscovered error.
3. A successful test is one that uncovers an undiscovered error.
4. The software confirms to the quality and reliable standards.
5. The tests are inadequate to detect possibly present errors.

##### 8.1.2. Testing Principles

1. All tests should be traceable to end user requirements.
2. Tests should be planned long before testing begins.
3. Testing should begin on a small scale and progress towards testing in large.
4. Exhaustive testing is not possible

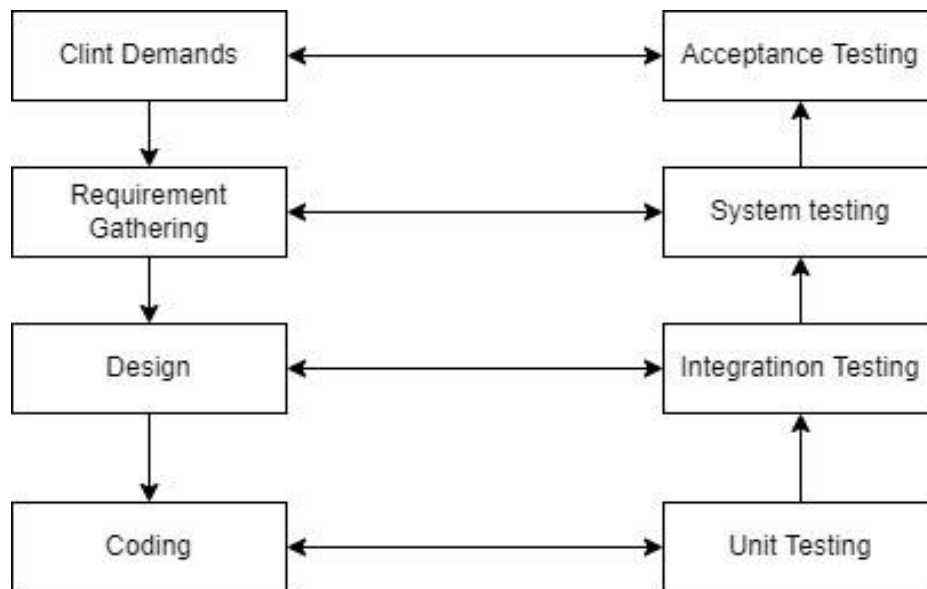
#### 8.2 Testing Strategies

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later during maintenance also.

##### 8.2.1. Psychology of Testing

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn't work.

### 8.2.2. Levels of Testing



Software Testing is an activity performed to identify errors so that errors can be removed to obtain a product with greater quality. To assure and maintain the quality of software and to represent the ultimate review of specification, design, and coding, Software testing is required. To uncover the errors, present in different phases we have the concept of levels of testing. The basic levels of testing are

#### 1. Unit testing

Unit testing focuses verification effort on the smallest unit of software i.e., the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins. Each module has been tested by giving different sets of inputs. When developing the module as well as finishing the development so that each module works without any error. The inputs are validated when accepting from the user.

#### 2. Integration Testing

After the unit testing, we must perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions. In this project the main system is formed by integrating all the modules. When integrating all the modules we have checked whether the integration effects working of any of the services by giving different combinations of inputs with which the two services run perfectly before Integration.

### **3.System Testing**

Here the entire software system is tested. The reference document for this process is the requirements document, and the goals to see if software meets its requirements. Here entire 'Color Coded Resistor Calculator' has been tested against requirements of project and it is checked whether all requirements of project have been satisfied or not.

### **4.Acceptance Testing**

Acceptance Test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system, the internal logic of program is not emphasized. In this project 'Color Coded Resistor Calculator' I have collected some data and tested whether project is working correctly or not. Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once. The testing phase is an important part of software development. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied

### **5. White Box Testing**

This is a unit testing method where a unit will be taken at a time and tested thoroughly at statement level to find the maximum possible errors. We tested step wise every piece of code, taking care that every statement in the code is executed at least once. The white box testing is also called Glass Box Testing We have generated a list of test cases, sample data. This is used to check all possible combinations of execution paths through the code at every module level.

### **6. Black Box Testing**

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

## 9. FUTURE ENHANCEMENTS

In future Enhancement we are planning and will implement the;

1. Adding more appearance
2. Making to visualize 3d model resistor
3. Scan the resistor automatically

Above mentioned points are the enhancement, which can be done to increase the applicability and the usage of this project





## 10.CONCLUSION

Resistor code colors are colored marks printed on resistors to display the electrical resistance and tolerance values. The resistor's physical value indicates the wattage rating. This project is very useful to find the resistor value and the tolerance, and it also calculate the temperature Coefficient. It helps us to calculate the multi band like 3,4,5,6-band and it can calculate the meg values also we can calculate very fast and its user interface is very easy to use and its user interface is very attractive and it is unique project



## 11.BIBLIOGRAPHY

### BOOKS

1. Jason Butler and Tony Caudill, "C#.NET Database Programming" Fifth Edition, Hungry minds, Inc. Publishing Company Limited, New Delhi, 2000.
2. Glen Johnson Wiley, Learning C#.NET, Fourth Edition, Hungry minds, Inc. Publishing Company Limited, Uttar Pradesh, 2001.
3. Company Limited, Uttar Pradesh, 2001.
4. Jason Buttre and Tony Caudil, "C#.NET Database Programming" Second Edition, Hungry minds, Inc. Publishing Company Limited, New Delhi, 2000.

### Website References

- [www.google.com](http://www.google.com)
- [www.youtube.com](http://www.youtube.com)
- [www.tutorialpoint.com](http://www.tutorialpoint.com)
- [www.freecodecamp.com](http://www.freecodecamp.com)

