Purpose of this document:

- To explore, and document, what other developers have in their:
  - Public GitHub repositories
  - Other online portfolios

- To come up with 4 or 5 portfolio project ideas:
  - Previous projects that I can polish and upload
  - New projects that need to be created for the portfolio

I will start by looking at some GitHub repositories:

- *(ProfSFrink/c_sharp_portfolio)*

  - Basic Math Test (tree/master/BasicMathTDDTest): Contains some basic maths functions and some methods to test them.
  - CodeFirstBDAssign (tree/master/CodeFirstDBAssign): An application to add "students" to a database using .NET framework
  - StudentMVC (tree/master/StudentMVC): MVC application to store and retrieve information about students.

- *(Silcott/PORTFOLIO-OF-PROJECTS)*

  - Legal Lead Closer (LegalLeadCloser): Legal application that collates client data and formats it.

  - Trackin!T (ISTA_Project/tree/master/myProject/Project_Track!T): multi-role, ticket application
    for generation and tracking of tickets. Tickets represent some entity or work that requires a track-able state and can be used to track the state of that entity.

  - Hunt the Wumpus - JB Version:

    Simple picture based adventure game build with "Monogame" and C#.

- *(bizzclaw/dotnet-portfolio)*

  This is a single MVC application build to showcase their dotnet skills. The user can create accounts and post blog posts as their user-account. There is a "How to run" section, and a "Design" section, in the README.md

After reviewing the portfolio websites visible on the forks of "emmabostian/developer-portfolios" what I have noticed:

- The sites are generally very modern and sleek (drop down menu and hover effects).

- The developers are predominantly front end.

- Many of the sites contain pages where they have showcase projects available

- The showcase projects tend to be fairly simple: (clone Wordle, Number guessing game, CSS Grid Visualiser)

The portfolio’s I looked at contain an array of different projects that demonstrate certain skills.

These can general be categorised into:

- C# Fundamentals
    - C# Syntax Proficiency
    - Application of OOP Principles
- Database Management
    - Integration with Entity Framework
    - Database Design and CRUD Operations
- ASP.NET MVC Development
    - Building and Designing MVC Applications
- Game Development
    - Monogame and C# Game Development
    - Graphical Game Design Skills
- Problem Solving
    - Creative Project Implementation
    - Troubleshooting and Bug Fixing

I think my portfolio should present a wide range of skills;
Whilst still being tailored to the area, or areas, in which I’m most interested.

The area’s in which I am most interested, and/or skilled, are:

- *Video Game Development*
- *Networking*
- *Software Design/Architecture*
- *Systems Design/Architecture*

Some general ideas for projects that I could put in my portfolio, that would display the necessary skills, whilst bolstering my own are:

- An MVC application
- A Database/SQL project
- Some other project to show fundamental principles of OOP (E,I,P)
- A Server/Client application for a simple game.
- A simple 2D single-player game with a high scores database.

An MVC application:

- My current idea is to build a desktop application that mechanics can use to keep track of what cars they have in, when they arrived, what work is being done, state of the work, etc.

- I would then have a simple web app that clients can enter their “SecureReferenceNumber” and it will serve them whatever information they are permitted to see.

A Database/SQL project:

- Most likely the MVC application above and the 2D single-player game I mentioned will have databases they use. Therefore this one probably isn’t necessary as a stand-alone project.

Some other project to show fundamental principles of OOP (E,I,P):

- The Client Server game application will contain all of these principles very clearly so perhaps that’s the best way for me to display that. This project feels like it’s only use would be to flesh out the size of the portfolio. Therefore, it’s probably not worth making as a stand-alone.

A Server/Client application for a simple game:

- My current idea for this project is to adapt a current project I have that uses C#’s “System.Net.Sockets” namespace to create a server client connection. It then uses a custom Packet system to communicate over the network.

- Will contain basic packets for movement of a player: jumping, picking up item.

- The packet system will be modular so it can be dragged easily from server to client to logon server etc. This creates an issue: What do we create to allow easy parsing of the packets when each system is required to perform different actions on the packet?

- To solve this the packet system will have a static “PacketParser” class. This is used to take the “Packet”, Header and Payload, and return defined objects of types that the “PacketHandler” class can use to preform the related action.

- The “PacketHandler” class will be abstract and contain virtual methods for handling all the default packets. This way, by default, the user can create an instance of the PacketHandler class and override the methods that they need to use to create the custom functionality they need for each application.

- The Client itself will just contain a simple 3D unity game with a player than can move, jump, interact with an entity and have these actions validated by the server before the game-state is updated.

A simple 2D single-player game with a high scores database:

- My idea for this game is to have a car drifting game with 1 or 2 tracks that the player can drive round and get a time and a drift score.
- The players can then enter their name to added to the high-scores which is stored in a database.
- The high-scores will then be displayed in their own window or after you've finished your lap of that track

Each of these projects will have a README detailing how to run/use the project and any other relevant information like dependencies or tools used.

They will also have a ".odt" or ".pdf" document made to detail the architecture/design and purpose of the project.

Relevant information to put in the README and/or design document:

- Project name
- Description
- Installation guide
- Usage instructions
- Technologies used
- Key Features
- Screenshots or Demos
- Code Structure explanation
- Contact Information
- Acknowledgments
- Status and Maintenance