**UnReaL Design Document**

# The Problem:

Long URL strings pose a significant issue for a multitude of reasons such as:

* They are difficult to read
* The lack of clarity leads to suspicions of spam or malware
* For apps such as Twitter, the character limit can present significant issues
* The links run a greater risk of being broken by user error
* They can undermine website rankings on search engines such as Google

This provides a gap in the market where a URL shortener could provide solutions for these issues via storage, encoding, decoding, and redirection.

I will aim to have a full working prototype in place in my git repository in under two days.

# Architecture:

## Proposed Tech Stack:

Due to my time travelling powers I have access to technologies from 2022, including ASP.NET Core which won’t be invented for another 10 years, which I suspect will give me an edge over the rest of the market. What .NET Core offers over traditional .NET is a greater portability as it is not restricted to only Windows machines.

With this in mind, I will develop an application using the following technologies:

* ASP.NET Core 3.1
* Entity Framework
* SQL Server

## Design:

My app will use an MVC (Model View Controller) design pattern, making use of interfaces and dependency injection for loose coupling and more granular testing. The following is my proposed class/view structure for the project:

**Controllers:**

* UnReaLController.cs

**Models:**

* ShortURL.cs
* ErrorViewModel.cs

**Views:**

* UnReaL/Create.cshtml
* UnReaL/Show.cshtml
* Shared/Error.cshtml

**Database:**

* UnReaLContext.cs

**Repository:**

* UnReaLService.cs
* IUnReaLService.cs
* BijectionService.cs
* IBijectionService.cs

In addition to this, I will have a localDb in the project files which should be able to run from wherever the software is hosted. Dynamically setting the root path for the project as part of the connection string should achieve this.

## Bijection:

Central to making this project work will be my implementation of bijection to enable the encoding and decoding of URL strings.

Bijection is a function between two sets, where all elements in one set are paired with exactly one element of the other set and no element is left unpaired.

This bijective mapping between two sets with cardinality is considered both injective and surjective, and so is also considered isomorphic (an isomorphism being a structure-preserving mapping between two structures of the same type that can be reversed by an inverse mapping).

This inverse mapping relationship is precisely what we need as with an appropriate reference alphabet and corresponding base, I can write an algorithm to effectively encode and decode URL strings for redirection to the URL’s intended location.

# Proposed Workflow:

