It began with one man’s dream: the dream of providing the ultimate key to everyone’s questions.

“That’s impossible!” They *all* told me, but dreams often aren’t so easy to kill. Especially not dreams so majestic as this one.

# Humble Beginnings

They say that even the longest journey begins with a single step, so I knew the key was to reduce every man’s journey into a series of single steps. When an epiphany hit me like a semi -truck with a bolt of lightning painted on the side: Every step we take is simply a decision!

My application makes decisions for you. It gives you the resources you need to take the steps of your journey on the path to the many grand adventures of your dreams.

The two primary goals I had before me were daunting:

* Ultimately Streamlined and Optimized Usability (USOU)
* Robust Enterprise Extensibility (REE)

To less of a dreamer, these goals might seem to be in conflict, but to me, their complementary nature was nothing short of magnificent.

# Robust Enterprise Extensibility (REE)

Extensibility is, by its very nature, random, so from the very beginning, the application is founded on extensibility, which provides the bedrock of this application’s very foundation as well as the center to its very core.

## …Speaking of Cores

“Random” is the name of the game in this application, so in order to be extensible, the primary “core” of random algorithms has been abstracted to allow for easy integration as the “science” of “randomness” (SoR) changes.

## Plugins

The program harnesses the power of “plugins” to delegate implementation of specific randomness algorithms. So as not to pigeonhole the development community with arbitrary constraints on the creation of these plugins, PluginFactories are used

**\*NOTE: PluginFactories must expose a parameterless constructor, and Plugins must have the appropriate PluginFactory identified by means of the PluginFactoryAttribute.**

## Plugin Repositories

For a truly extensible solution, I didn’t wish to hamstring the development community by forcing them to necessarily provide *files* with their plugins, so I have provided an abstract means of obtaining these “repositories” of plugins. Once more, I wanted the developer to be limited only by his imagination (and a simple interface definition) with regard to these repositories, so PluginRepositoryFactories provide a level of abstraction that sets the developer free

**\*NOTE: PluginRepositoryFactories must expose a parameterless constructor, and PluginRepsoitories must have the appropriate PluginRepositoryFactory identified by means of the PluginRepositoryFactoryAttribute.**

## Plugin Pickers

Many developers would have considered such a design sufficiently robust, but not all developers are big dreamers. To provide for a truly enterprise experience, abstract plugin pickers tie the whole thing together and pick (at random) a plugin from the repository provided to use to solve the question posed to the program. Because Plugin Pickers are such a key component to the overall process, they are a bit more constricted than plugins and repositories. Only advanced programmers should attempt extending these objects.

## The User Experience (TUE)

The User Experience (TUE) is important to every user’s experience, and every user’s experience with TUE is a difference experience. So to be flexible, the TUE has been “abstracted” so that programmers can customize the program to their liking. Most users enjoy graphical user interface (GUE), so one has been included.

# Ultimately Streamlined and Optimized Usability (USOU)

At the heart of the process, “numbers” are “randomly” “generated” through a vast and complex “engine.” In professional circles, this “engine” is sometimes called a “Random Number Generator,” but I didn’t want esoteric lingo to frighten users, so I streamlined that term to simply “Rand-umber-ator,” a term that’s so simple and catchy, I anticipate it becoming a household name in the next week or so….right about the time every household on the planet begins comfortably turning to my program to help people take the first steps of their next journey

… to fulfill their dreams.

# Technical Details

## Randumberator.Core

As its name suggests, this namespace serves as the core to Randumberator. It includes the base definitions for plugins, etc. as well as “FailSafe” implementations. It includes a static “Engine” class which provides access to the applied “Random Algorithm Core” implementation.

## Plugins

The applied Plugin is randomly selected in “Real Time,” from the collection of available plugins. The following stock plugins (as well as their factories) are provided:

* **Randumberator.Plugin.Always0** – You wouldn’t expect a RNG to return the same value every time, so the sudden appearance of such a creature is nothing if not surprisingly random!
* **Randumberator.Plugin.Always1** – A lot like the Randumberator.Plugin.Always0 plugin…only the complete opposite. (Now that’s random!)
* **Randumberator.Plugin.Random** – If it’s good enough for programmers everywhere, then it will suffice…in a pinch. This plugin leverages the awesome power of the native .Net framework to determine a random number.
* **Randumberator.Plugin.SuperRandom** – When Random’s not quite random enough, this powerhouse seeds the randomizer with a large number (32 whopping bits!)
* **Randumberator.Plugin.Random64** – For those truly difficult questions, Random64 expands on the SuperRandom’s power by supporting 64-bit integer seeds. (It’s roughly ***twice*** as random as Super Random!)
* **Randumberator.Plugin.NeverTheSameTwice** – Provides for a truly random experience by logging the most recent value to ensure that answers are “Never The Same Twice” (patent pending)
* **Randumberator.Plugin.Ramdom** – That’s no Typo! This plugin delves into the ever-changing jungle of RAM usage to algorithmize a truly random value so you don’t have to!

## Plugin Repository

There’s been a lot of talk about this “Cloud” computing stuff, which sounds perfect for a big dreamer like me, but I have not had time to look into that, so there is currently only a FileSystem Repository (and factory). The framework is in place, though, for members of the development community to create repositories using virtually any technology (e.g. XML, RSS, Cloud, Azure, OCR, Neural Networks, Biometrics, Pintrest, iTunes, Google Glass, etc.)

## Plugin Picker

Each time a “question” is posed, the Plugin Picker randomly “picks” a plugin from the repository it’s been given in order to determine the best “answer.”