Requirements:

1. Respect Robots.txt
   1. Optionally ignore them
2. Start with one or multiple seed links
3. Allow plugins to control the rules for what pages get added to the Queue to crawl
4. Allow plugins to control what happens with the data as the pages are crawled
   1. For Example/Ideas:
      1. Log all URLs containing a specific word
      2. Log all URLs with a specific <title>
      3. Attempt to harvest e-mail address, steam keys, or other data
      4. Attempt to purchase a product or check it a product is in stock
      5. Alert by e-mail/text message/MessageBox or other means when target is found
5. Store the crawling queue in a database with last date crawled
   1. **Decision Required**: Dapper, Entity Framework, something else?
   2. Possible support for OR make it easy to add support for multiple DBs
      1. All DB access though an interface
   3. Store the last crawled date and URL in the DB
      1. **TODO:** Decide what else the DB should track
6. **Decision Required**: User interface
   1. The core logic should be in a separate library from the UI
   2. Support one, some or all:
      1. Console UI
      2. WinForms UI
      3. WPF UI
      4. Web UI (ASP.NET/Blazor)
7. Target .NET 5
8. Crawl on multiple threads
9. Implement a customizable delay between crawling on the same domain
   1. This may possibly be found in the robots.txt
10. Allow the user-agent to be customized

TODO:

1. Name for the project:
2. Determine classes and database design.

To Learn:

1. How to do plugins
2. How to do threading
3. More about HTML/DOM/CSS
4. More in depth parsing with HtmlAgilityPack

Learning Resources:

<https://www.codeproject.com/Articles/1087859/Web-crawling-with-Csharp-part-one-8>

<https://html-agility-pack.net/documentation>

<https://en.wikipedia.org/wiki/Crawl_frontier>

<https://en.wikipedia.org/robots.txt>

<https://www.contentkingapp.com/academy/crawler-traps/>

**Classes:**

**GlobalConfiguration**: (Store global configuration data)

* IDataConnection connection; (Connection to the database)

**RobotParser**: (Parse robots.txt rules)

* Parses the rules and creates an instance of a model representing them

**Crawler:** (Main crawling logic)

* Will need rules for crawling
  + Robot, Rates, etc