**Crypto Info Service – Blazor Web Application**

Eldin Delic Hodzic – X00153243

**Description:**

The API that I have chosen is a crypto-to-crypto exchange service called Nexchange. The API displays information such as different types of currencies, pairs and price information such as history and rates. The web application contains 2 different pages:

* Currencies: displays different currencies, if they are a crypto currency, minimum and maximum amount of the currency, and pair compatibility
* Price History: displays the creation and expiration of any currency pairs, asking and bidding price, rates and market

**Functions:**

Every web page has a different function and I have implemented a total of 3 simple functions:

* Calculation button
* Search form
* Sorting function

There calculation button ‘total minimum confirmations’ counts the total of minimum confirmations of all pair currencies.

The search function allows a user to search for a pair, the number of hours and the number of rows to display (pair name, hours, data points). There is validation for each of the fields. The pair name field ONLY takes strings where the minimum length is 5 and the maximum length is 12. This is because no pair name has a number in it and the smallest pair name is 5 characters and the longest pair name is 12 characters long.

The sorting function allows the user to sort the data based on the different options. The different options include ID, base rate, pair name, asking fee, coin names and minimum/maximum amount.

**Testing:**

There was only 1 component I could really test in this application and that was the search form. When I created the search form, I realised that there weren’t any applicable built-in validations for the type of form I had so I had to create my own custom validation since there was a lot of validation for the form.

// The class below is the custom validation function

// Custom validation class

public class CheckPairNameValidationAttribute : ValidationAttribute

{

// Custom validation function

protected override ValidationResult IsValid(object value,

ValidationContext validationContext)

{

// Converts value into a string

string valueString = Convert.ToString(value);

{

// Checks if the length of the value is in between 5 and 12 and contains letters only

if (valueString.Length > 5 && valueString.Length < 12 && valueString.All(char.IsLetter))

{

return ValidationResult.Success;

}

}

// if the value does not pass above, sends error message

return new ValidationResult("Enter Valid Pair Name from the Pair List",

new[] { validationContext.MemberName });

}

}

The form only allows integers between 1 and 24 for the hours field and it does not allow any other value type, if the user tries to enter a different value type, it resets the field to it’s default value, 0. This is the same for the data points field except the range for the data points is 1-10.

Other small tests included verifying if the calculations were correct for the total minimum confirmations, in which case the calculations were correct.

Checking if the sorting function actually sorted the data based on what the ‘Sort by’ value is, in which case it did sort the data correctly.

**URIs:**

**GitHub:** https://github.com/eIdino/CryptoService.git

**Azure Hosted App:** https://cryptoservice20211208182759.azurewebsites.net/

**VIDEO LINK:** https://streamable.com/pu26e1

**Screenshots:**

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated