# Documentation

# Sahas Gunasekara

20462075

# Part 1

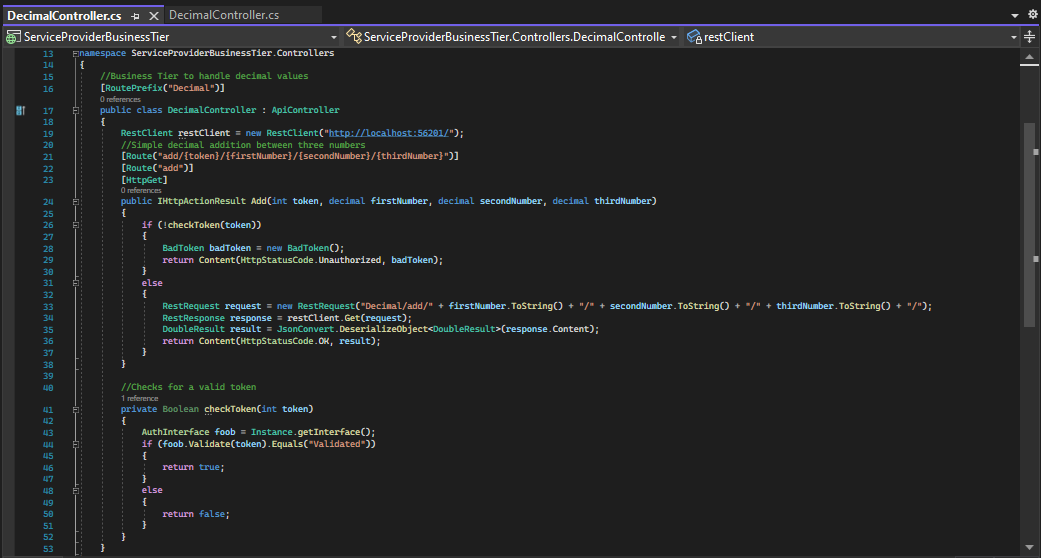
When adding a new Service/Functionality to the program there are a few key steps we need to follow. You could either add a new functionality as a new Controller altogether or you can add it as a function to an existing controller with its own unique route. Here’s an example of us adding a Controller to handling three Decimal inputs. We add it to our ServiceProvider API.

Text

Description automatically generatedText

Description automatically generatedDecimalController.cs added, as a new Web API 2 Controller. We use Get and URL Passing and the Code should be added as follows. We add a RoutePrefix and then added a Route with the parameters as the given example. If the input operands are of type Decimal, then we create a DoubleResult object and return it to the BusinessTier or else it would be of type IntResult. All these methods should be of type GET since we pass in a few values and receive a value from the API.

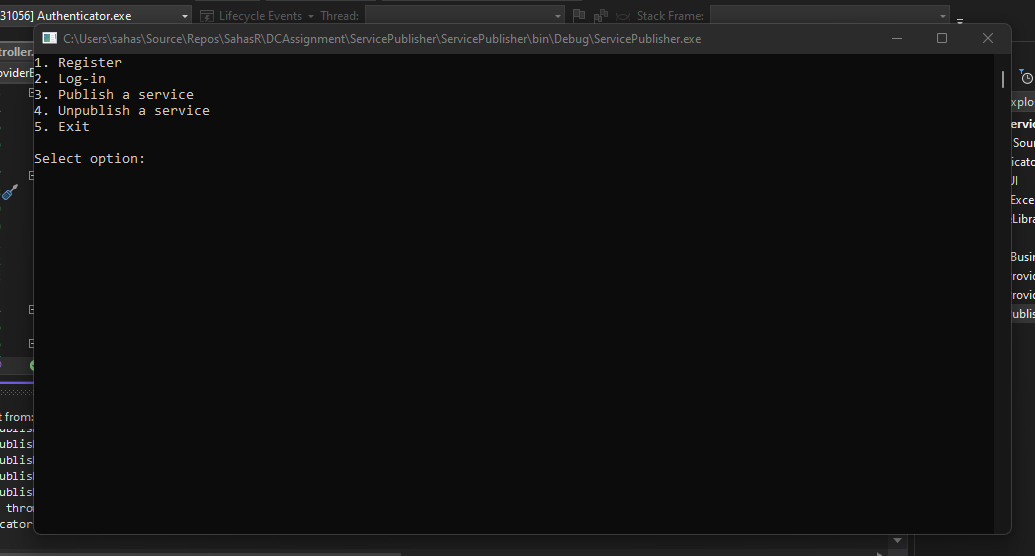
After this is done, we move towards the Business Tier. This is the area in which our client will be talking to. The Business Tier is tasked with passing on the parameters to the DecimalControllerAPI and checking for the token passed through.

In the ServiceProviderBusinessTier API we add another controller which takes in the token and the parameters. We must connect to the ServiceProvider API as well so we must keep a check of the ClientURL and the Route to call it to pass functionality onwards.

Here we call a function checkToken() with the passed token from the URL, which validates the current token to ensure that trusted clients are only connected. After which we either passthrough the parameters as we have received except the token or return a BadToken object which contains JSON values to inform the user it was an unauthorized transaction as per the assignment documentation. If the StatusCode is okay we passthrough the content received by the ServiceProviderAPI, back to the client.

After the code has been added we should register it as an actual service to be allowed access inside the Registry. For this we need to debug the Instances of

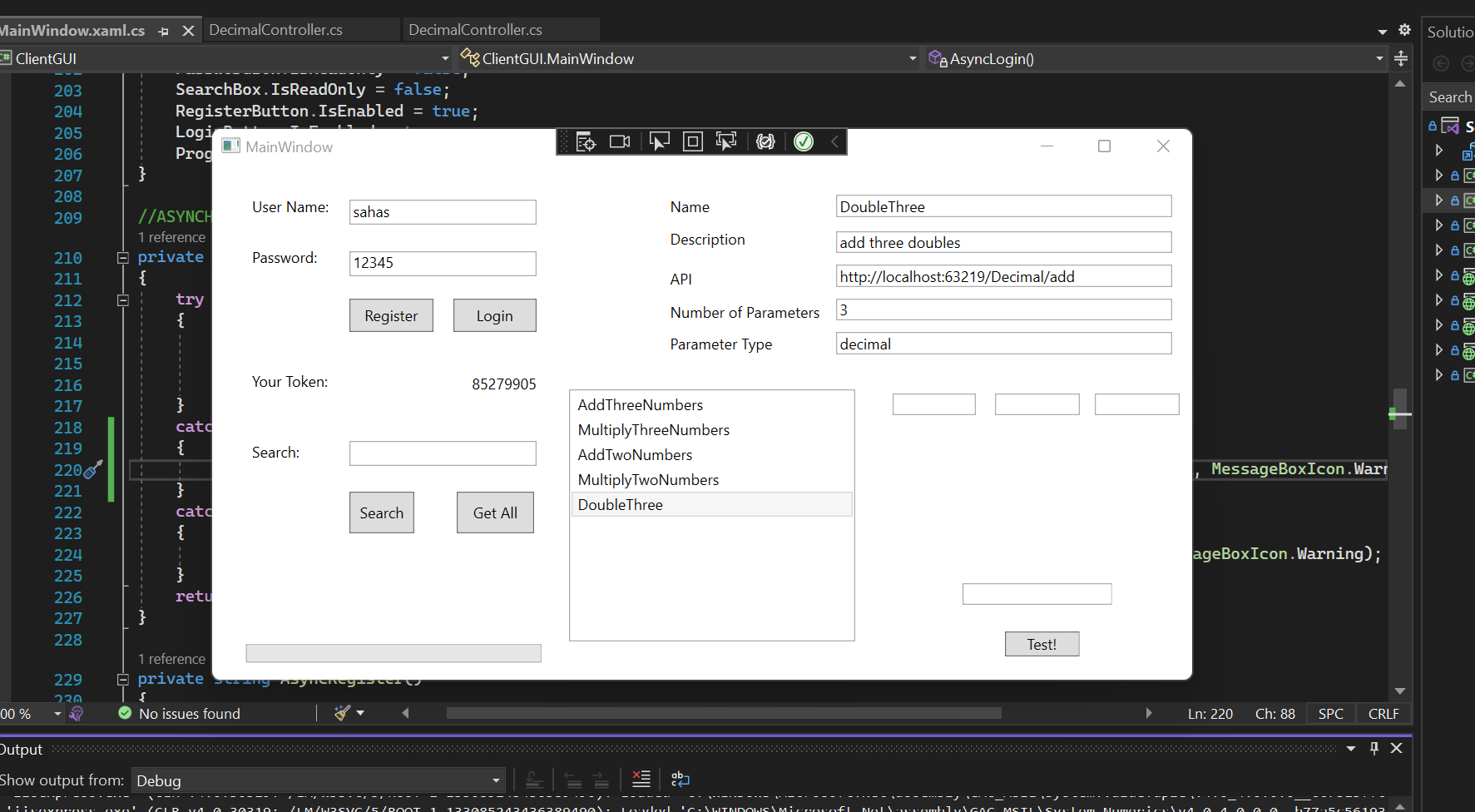
1. Authenticator -> To Authenticate the User who is logged into the ServicePublisher
2. Registry -> API that handles the registrations and management of services.
3. RegistryBusinessTier -> Business Tier API that handles the Registry API.
4. ServicePublisher -> The Console Application we use to register a new API.

The instances should ideally be started in the previous order however, ServicePublisher should be the last as it depends on Authenticator to gain access and Registry APIs to manipulate.

You need to first log-in (if you don’t have an account just enter option 1 to register first), and then go to publish a service option in which you will enter the accurate details such as Service Name, Service Description, API Endpoint (eg: <http://localhost:33219/Decimal/add/>), Number of Variables and Type of Variables (decimal or integer). After which it will be saved in a local text file in JSON Format as the following.

Text

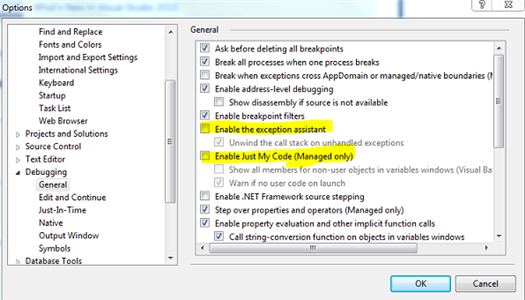
Description automatically generatedThis means that the service is successfully added and registered which means you can now access it through the Client!

All you have to do is when you open up the client is to login (or again register if you don’t have an account) to get a valid token. Then either search for this service by name or fetch all and select from the ListBox Menu at which point you will be able to dynamically get the number of textboxes which you will be able to add all the values for. There will be format checking for either option depending on whether you select Decimal or Integer.

PS: There was an incident where we received an error of System.ServiceModel.FaultException `1 when handling FaultContracts where we got a error of not handling the Exception we threw whereas it was perfectly alright and we did catch it. Therefore, for the program to work we had to follow the documentation on this resource:

<https://social.technet.microsoft.com/wiki/contents/articles/17418.the-famous-system-servicemodel-faultexception1-was-unhandled-by-user-code.aspx>

Which asks us to disable the Exception Assistant and disable the Just My Code Option from the Option Menu of the IDE, in the Debugging -> General Section for the program to work without any hiccups.



# Part 2