**Introduction to Project**



**HackOCompile,** a project which is named due to its capabilities. HackOCompile is named by the addition of three abbreviations where Hack means HackerEarth, O refers to Online and Compile refers to Compiler, when pronounced without abbreviations, it is referred as HackerEarth Online Compiler.

This is a very powerful .NET project capable of compiling and running 12 programming languages which are C, C++, C++11(ANSI), Clojure, C#, Java, JavaScript, Haskell, Perl, PHP, Python, Ruby. This project is fully compatible with Windows OS and uses only 15 MB of RAM and 3 MB of hard disk space and an active internet connection.  
  
As this is an Online Compiler all the execution and compilation is done on a remote server unknown to the user, physically these servers are situated in HackerEarth Data Center at Bangalore, but this doesn’t matters where the servers are situated as for an end user it won’t be much useful.

The project is fully coded in C# and in Visual Studio 2013 containing nearly 680 lines of code.

**Objectives**

The primary objective of this project is to give Developers the freedom to switch to 12 programming languages at one go without installing any SDK (Software Development Kit).

For Instance: If a client gives a Developer a project which is very unique and never done before, the developer has to first create an algorithm to work on the project given to check whether all the logic he has created works or not. Supposedly, he tests the algorithm in Python programming language as it will be the fastest way to test an algorithm, he first has to install Python SDK from internet which is very time consuming and will require further client side customizations to run the Python SDK. Then, he will apply that algorithm on a language he likes the most, thus installing an SDK for the same, resulting in a very low production time.  
  
By this project he can directly select Python language and develop an algorithm to test, and then use that algorithm in different language of its choice without installing any SDK or client side customization to run those SDK’s.

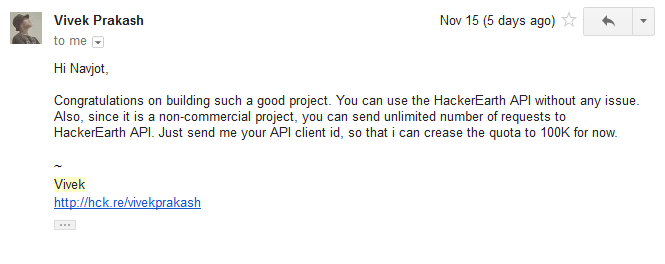
So, this project can be very useful in development of many products and software solutions.

**Project Scope**

This project can be very useful in the development of many products and software solutions as it will help many developers to run and test their codes and algorithm. Furthermore this project is now published on Github repository as an **Open-Source Project**, giving developers another freedom to customize the project for their own use and easiness.

In a recent talk with the owner of **HackerEarth’s Founder Mr. Vivek Prakash**, I asked him to review this project and check whether this project is under their Policies and Regulations. The response automatically shows the scope of project in the IT Industry.

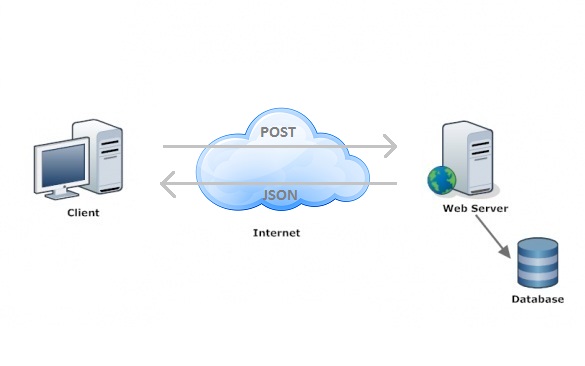
The normal requests by any software to the HackerEarth API’s are only limited to 1000 requests, but as considering the scope of this project they allowed my project to give Unlimited requests to their servers.



**Methodology Used**

This project uses HackerEarth’s API (Application Programming Interface) to compile and execute programs. The methodology used here is sending POST requests to HackerEarth’s servers and getting JSON (JavaScript Object Notation) requests in return. There are some predefined structures of API POST requests that have to meet the requirements to get response in return.

Flow of POST and JSON response requests can be understood by the below diagram:



The requirements for the HackerEarth’s POST API are:

**Endpoints**

http://api.hackerearth.com/code/compile/

http://api.hackerearth.com/code/run/

**Client Secret**

client\_secret is a 40 character random key that serves as an identification for a particular client and must be provided when communicating with the API. You can create a client\_secret on the API registration page. Failing to provide a client\_secret returns the following 400 Bad Request response.

Python Code:

>> curl -i http://api.hackerearth.com/code/compile/

>>

HTTP/1.1 400 BAD REQUEST

Date: Wed, 5 Nov 2014 22:56:25 GMT

Server: Apache/2.2.22 (Ubuntu)

Connection: close

Transfer-Encoding: chunked

Content-Type: application/json

{"message": "ArgumentMissingError: client\_secret is needed!", "errors": {}}

Providing an invalid client\_secret returns the following 400 Bad Request response. The invalid client\_secret is also returned in 'errors' attribute of the JSON response object.

Python Code:

>> curl -i --data "client\_secret=adsafwqf5dcb5757e72c96d9" http://api.hackerearth.com/code/compile/

>>

HTTP/1.1 400 BAD REQUEST

Date: Wed, 23 Jan 2013 23:00:20 GMT

Server: Apache/2.2.22 (Ubuntu)

Connection: close

Transfer-Encoding: chunked

Content-Type: application/json

{"message": "UnregisteredClientError: Client does not exist", "errors": {"client\_secret": "adsafwqf5dcb5757e72c96d969a1a"}}

**Source**

source is the actual text or the source code that you want to compile or run. Failing to provide the source parameter returns 400 response.

Python Code:

>> curl -i --data "client\_secret=5db3f1c12c59caa1002d1cb5" http://api.hackerearth.com/code/compile/

>>

HTTP/1.1 400 BAD REQUEST

Date: Thu, 24 Jan 2013 16:38:09 GMT

Server: Apache/2.2.22 (Ubuntu)

Connection: close

Transfer-Encoding: chunked

Content-Type: application/json

{"message": "ArgumentMissingError: source is needed!", "errors": {}}

**Language**

lang is a string representing the programming language in which the source is written. lang is a required parameter and a similar response as that of source is returned by the API if lang is missing in the POST requests.   
Possible values of lang argument are C, CPP, CPP11, CLOJURE, CSHARP, JAVA, JAVASCRIPT, HASKELL, PERL, PHP, PYTHON, RUBY. The corresponding langauges are C, C++, C++11, Clojure, C#, Java, JavaScript, Haskell, Perl, PHP, Python and Ruby.

**Input**

input is an optional parameter which represents the input to be fed to your program.

**Time Limit**

time\_limit is an optional parameter which is the max time for which program is allowed to run. Maximum value of time limit can be 5 seconds. Any value greater than 5 in the API request is set to 5. Exceeding the time limit returns the status as *TLE* in the response.

**Memory Limit**

memory\_limit is an optional paramer which is the max memory allowed to be used by the program. Maximum value of memory limit can be 262144 (256 MB). Any value greater than 262144 in the API request is set to 262144. Exceeding the memory limit returns the status as *MLE* in the response.

**Asynchronous Requests**

async is an optional parameter which is the API request mode used by the program. The value 1 turns it on. You should use async mode while writing applications so that you can avoid processes from hanging for response and reduce memory overhead and timeout.

**Example of Successful POST request:**

For Compilation:

http://api.hackerearth.com/code/compile?lang=C&source=%23include+%3Cstdio.h%3E%0A%0Aint+main%28%29+{%0A++++printf%28%22Hello%22%29%3B%0A++++int+n%3B%0A++++scanf%28%22%25d%22%2C+%26n%29%3B%0A++++printf%28%22\n%25d\n%22%2C+n%29%3B%0A++++return+0%3B%0A}%0A&callback=http%3A%2F%2Fexample.com%2Freceive-hackerearth-response%2F&async=1&time\_limit=5&client\_secret=4df77c2c2eb62f9adb20bd1127f6f44a4ce6cda4&id=123&memory\_limit=262144

For Execution:

http://api.hackerearth.com/code/run?lang=C&source=%23include+%3Cstdio.h%3E%0A%0Aint+main%28%29+{%0A++++printf%28%22Hello%22%29%3B%0A++++int+n%3B%0A++++scanf%28%22%25d%22%2C+%26n%29%3B%0A++++printf%28%22\n%25d\n%22%2C+n%29%3B%0A++++return+0%3B%0A}%0A&callback=http%3A%2F%2Fexample.com%2Freceive-hackerearth-response%2F&async=1&time\_limit=5&client\_secret=4df77c2c2eb62f9adb20bd1127f6f44a4ce6cda4&id=123&memory\_limit=262144

**Example of Successful JSON response:**

{"errors": {}, "code\_id": "3d255bX", "id": 123, "message": "OK", "run\_status": {"status": "NA", "time\_limit": 5, "async": 1, "memory\_limit": 262144}, "compile\_status": "Compiling...", "web\_link": "http://code.hackerearth.com/3d255bX"}

This project uses Newtonsoft JSON parser to parse the above JSON string to Objects, the object notation after parsing is:

{

"errors":

{

},

"code\_id": “3d255bX",

"id": 123,

"message": "OK",

"run\_status":

{

"status": "NA",

"time\_limit": 5,

"async": 1,

"memory\_limit": 262144},

"compile\_status": "Compiling...",

"web\_link": <http://code.hackerearth.com/3d255bX>

}

}

**Example of Unsuccessful JSON response:**

{

"errors": “4:1: error: expected ';' before 'int'”,

"code\_id": “3d255bX",

"id": 123,

"message": "OK",

"run\_status":

{

"status": "NA",

"time\_limit": 5,

"async": 1,

"memory\_limit": 262144},

"compile\_status": "Compiling...",

"web\_link": <http://code.hackerearth.com/3d255bX>

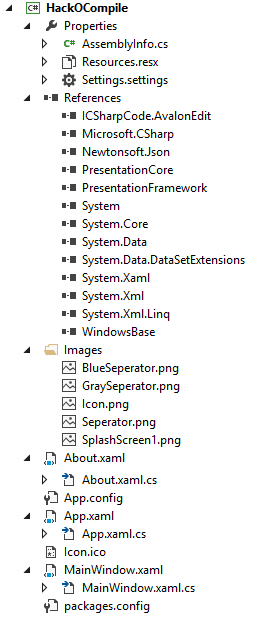
}

}

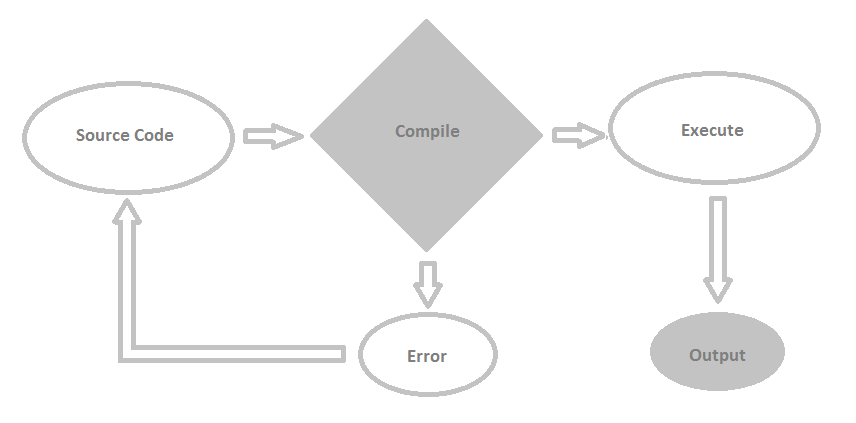
**Modules Developed**

The module structure of the project contains two Windows having a front end code for managing the layout or user interface of the project and a backend c# code to execute the events occurred in the project, except this there is one SplashScreenImage which displays before the starting of the project and some images to make the project User Interface more attractive.

The module structure is:



**Data Flow Diagram**



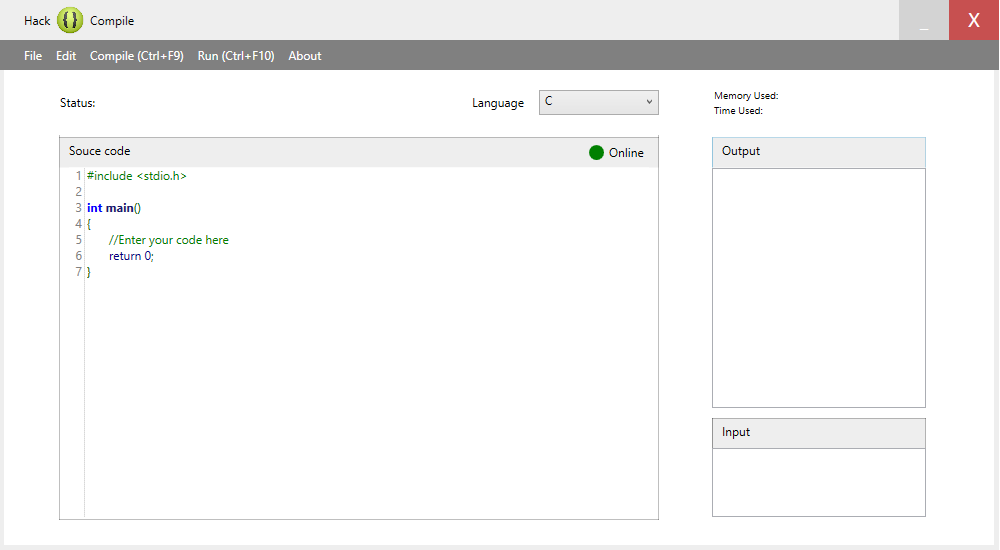
The data flow diagram is same as other compilers, the only difference is all the compilation and execution will take place online. The rest features like compilation (CTRL+F9) and execution (CTRL+F10) are same like other SDK’s to make it more easy to remember product as their won’t be something new to learn.

**Snapshots**

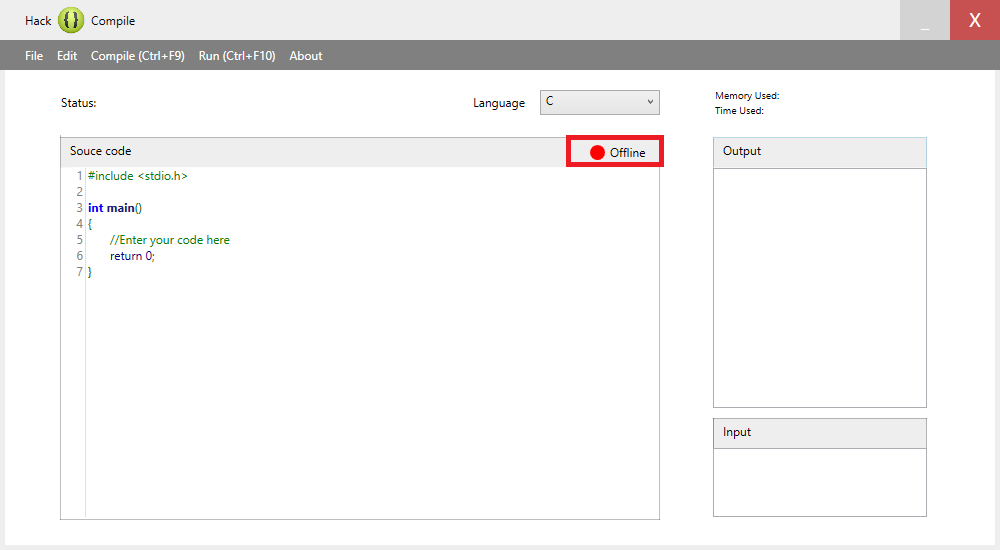
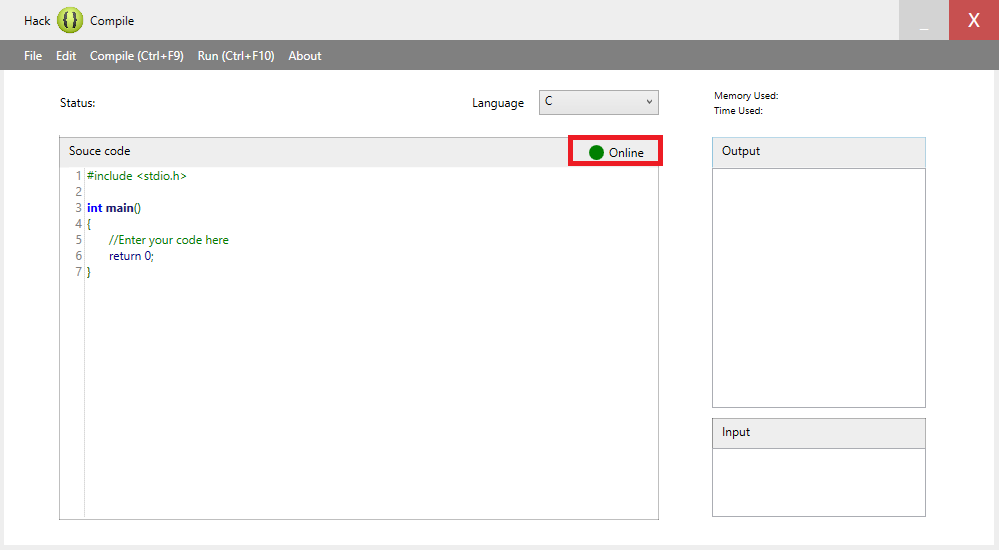
1. **Loading of Project**



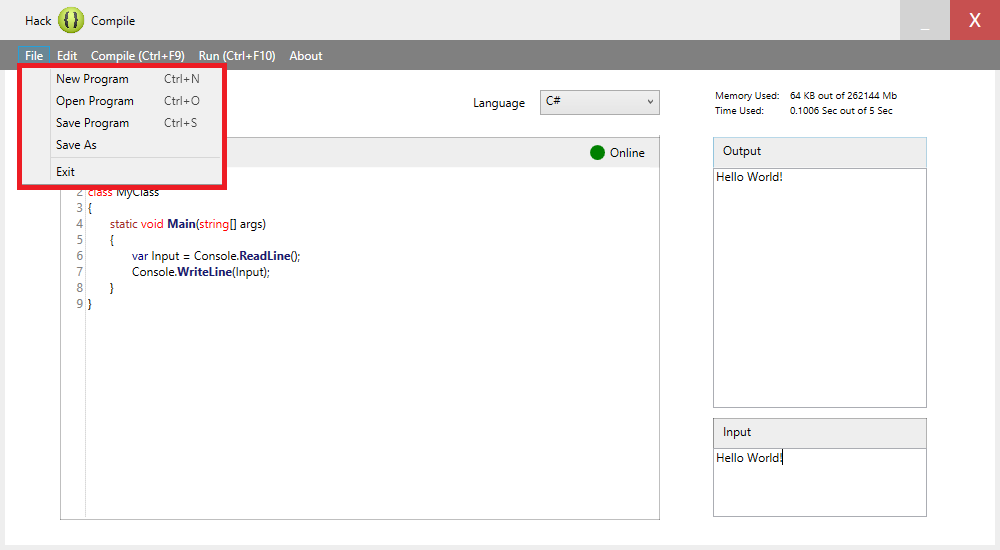
1. **Home Screen**



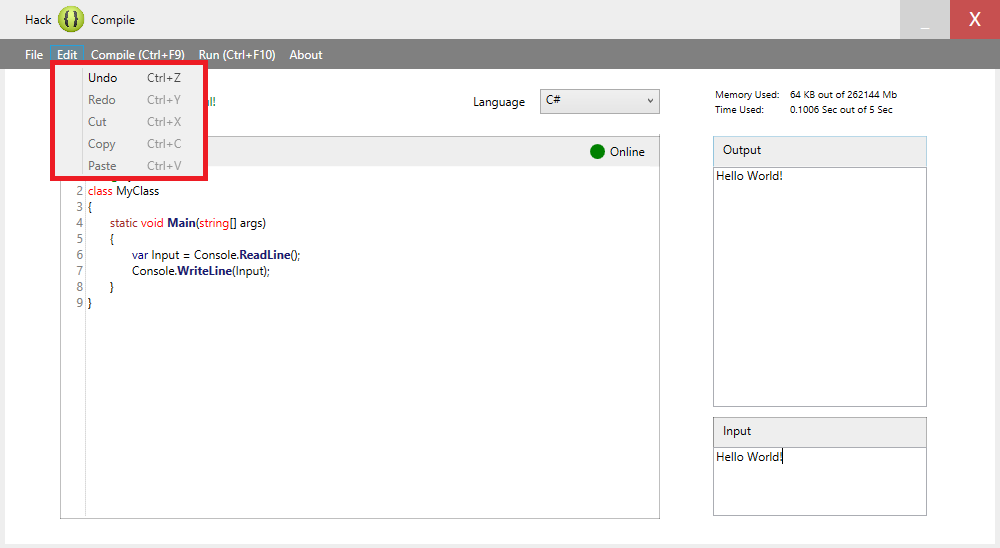
1. **Online/Offline Status**



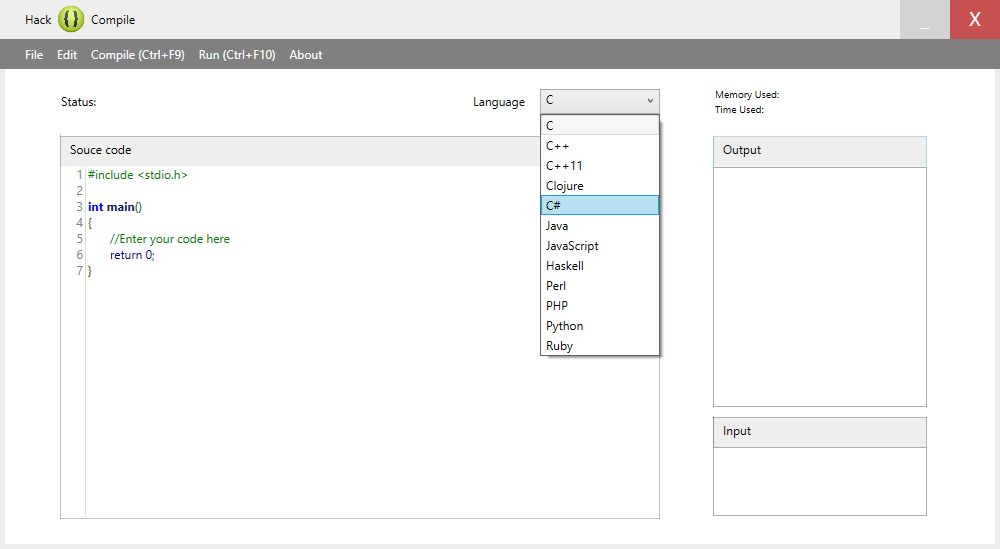
1. **File Menu**



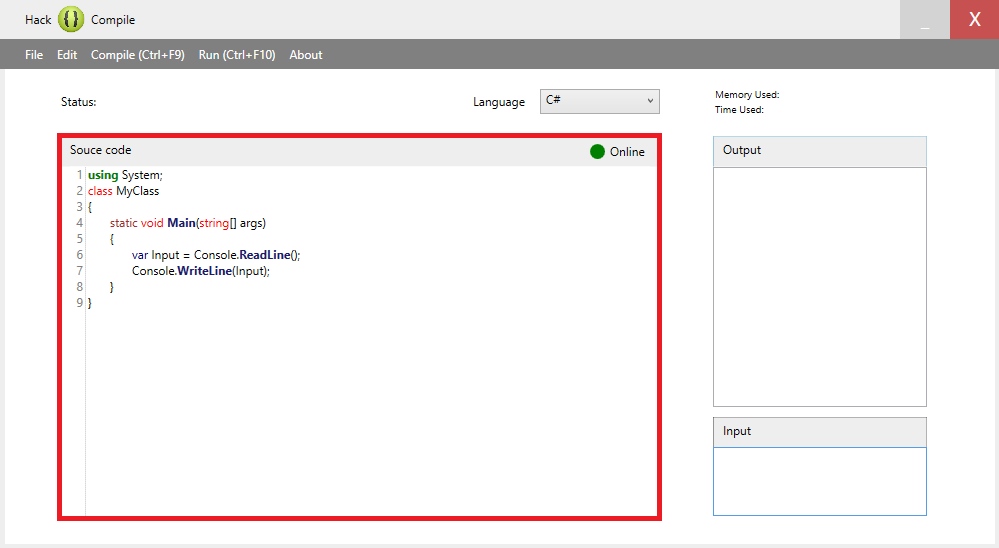
1. **Edit Menu**



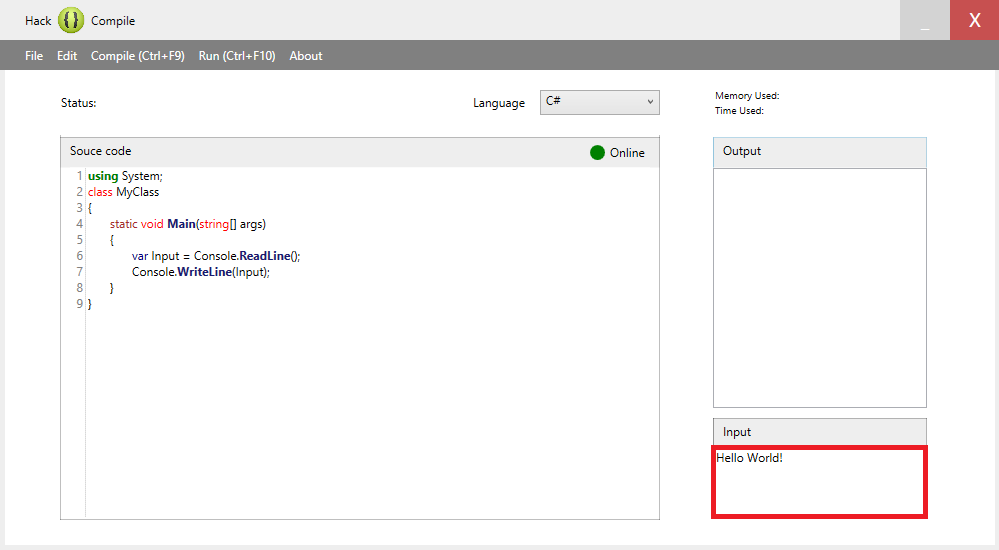
1. **Selecting Language**



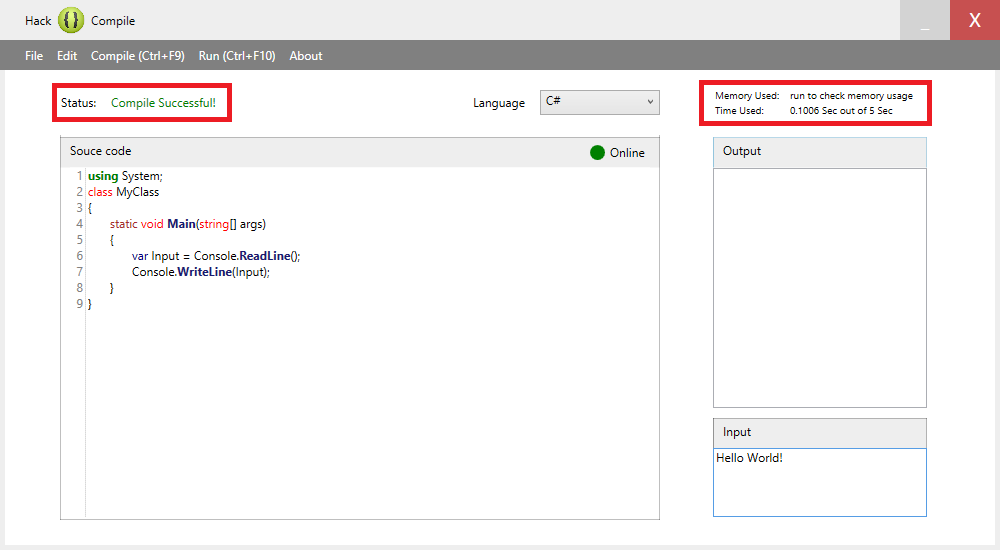
1. **Entering Source Code**



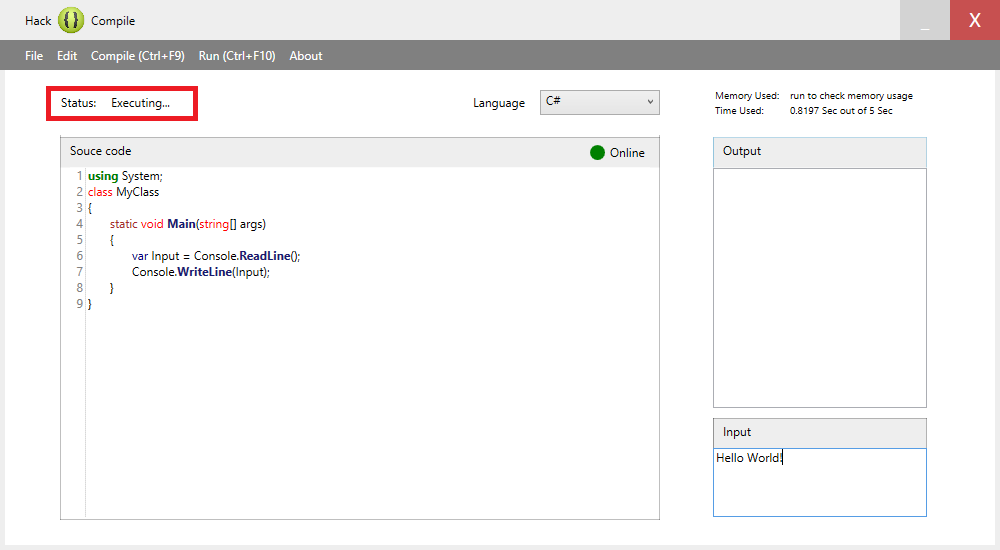
1. **Entering Input**



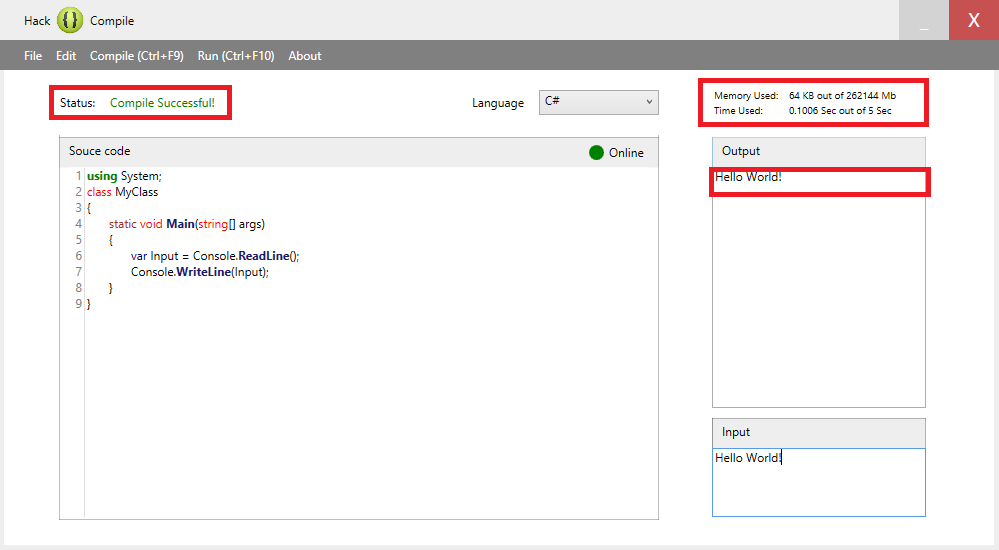
1. **Compiling (CTRL+F9)**



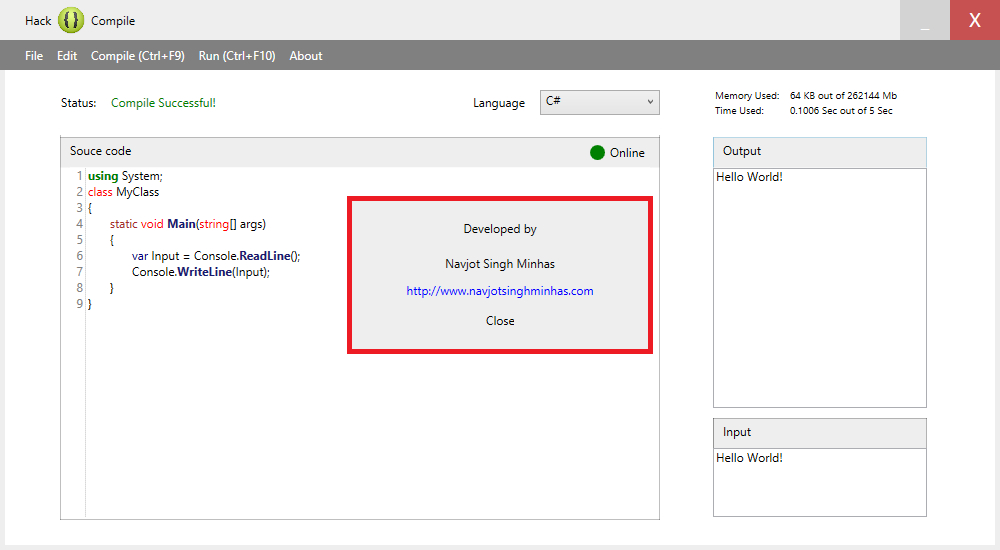
1. **Executing (CTRL+F10)**



1. **Output**



1. **About Menu**



**Future Scope and Conclusion**

The project can be used in colleges and schools to educate students, as most of the colleges or schools have outdated SDK’s installed, which effects students in many ways. By using this project we can be sure that the SDK’s are up-to-date without even updating the college or school computer labs.  
  
This project can also be used by developers for producing efficient codes without installing SDK’s which are nearly 4 to 8 GB in space and requires very efficient hardware whereas this project can do the same task in very less (~ 15 MB RAM and 3 MB Disk Space).

If any student or any developers thinks that the UI is not suitable for him to work on, or is having problem in understanding the full functionality or want to know the logic behind the project development, he can always see the source code of the project on GitHub and can make it more efficient to use.

As a result this project can be used in many ways to make the work and education more beneficial.

**References**

1. <http://developer.hackerearth.com>
2. <http://www.stackoverflow.com>
3. <http://images.google.com>