MRZ Verifier Manual Guide

V1.0

By Cliff Mak

30 April 2018

Table of Contents

[Introduction 3](#_Toc512966889)

[Architecture 3](#_Toc512966890)

[Features and Snapshots 4](#_Toc512966891)

[Addendum 8](#_Toc512966892)

[Invoking Web API Externally via Add-on Tool 8](#_Toc512966893)

[Log Checking 9](#_Toc512966894)

[MS Unit Test Passport Validator DLL 10](#_Toc512966895)

[Link References 11](#_Toc512966896)

# Introduction

Most travel passports worldwide are Machine Readable Passport (MRP). They are standardized by the ICAO Document 9303 (endorsed by the International Organization for Standardization and the International Electrotechnical Commission as ISO/IEC 7501-1) and have a special machine-readable zone (MRZ), which is usually at the bottom of the identity page at the beginning of a passport. This application, namely MRZVerifier, aimed to verify the given user input with the MRZ data. This application adapted distributed service architecture in which the frontend interface and backend service are separated. MRZVerifier is a MVC WebAPI acts as backbone while MRZFrontDesk is a MVC Website operates as a single page application.

# Architecture

The MRZVerifier system comprised:

* MRZFrontDesk
  + A frontend web application to collect data for validating and displaying result.
* MRZVerifier WebAPI
  + A RESTful web service that enable frontend calls to validate data.
* PassportValidator
  + A reusable class library (DLL) that furnished passport validation logic.

MRZ Verifier WebAPI

MRZ FrontDesk

request

response

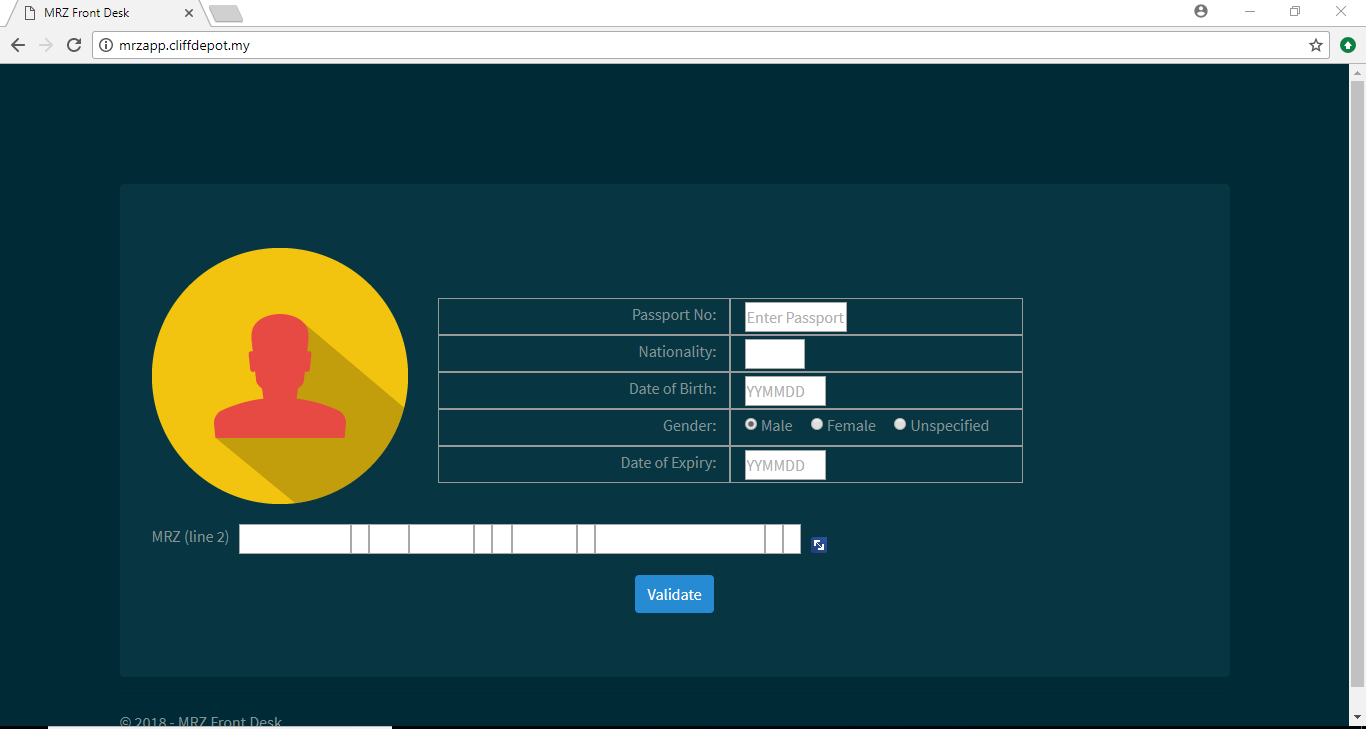
PassportValidator DLL

Endpoint URL:

* MRZ Verifier WebAPI - <http://service.cliffdepot.my>
* MRZ Frontend - <http://mrzapp.cliffdepot.my>

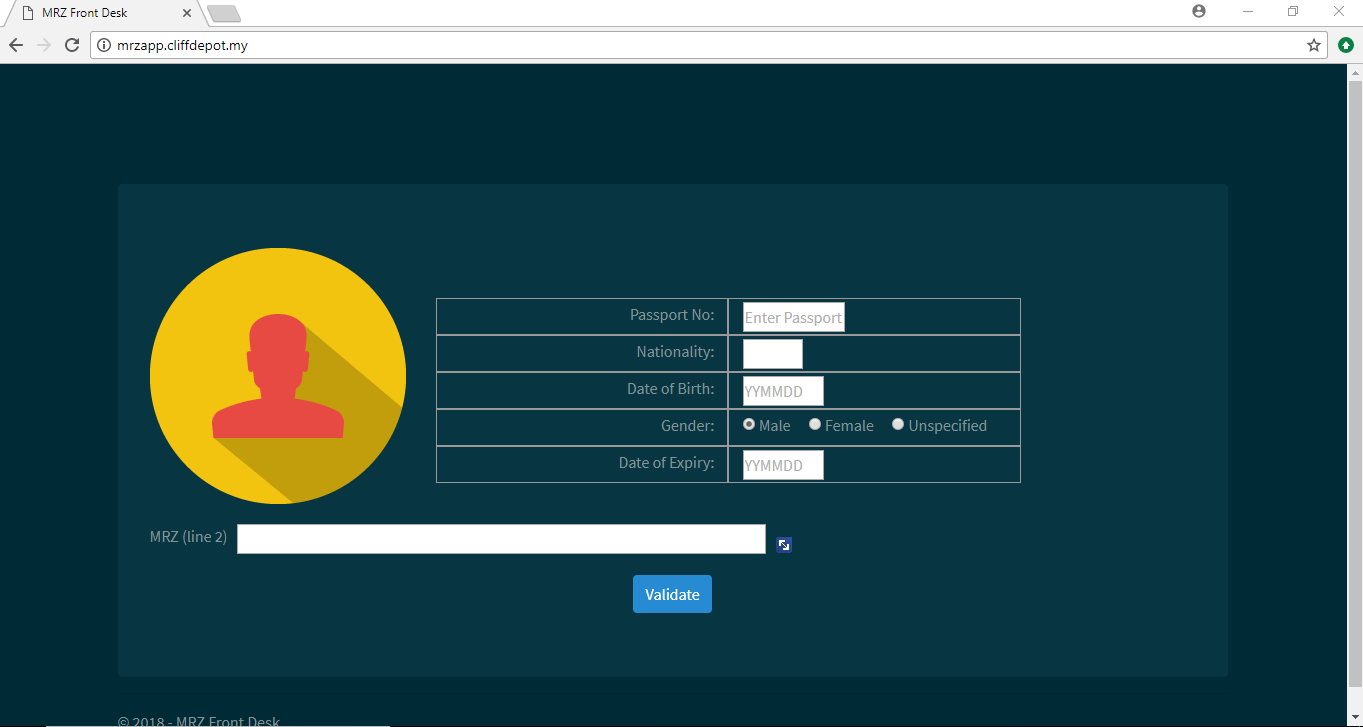
# Features and Snapshots

The landing page will be “../Home/Index”, by default it will auto-route when user go to *http://mrzapp.cliffdepot.my*. User is required to provide passport number, nationality, date of birth, gender, date of passport expiry and MRZ line 2 data. These inputs are compulsory fields. Validation is performed both at frontend application and backend API web service.



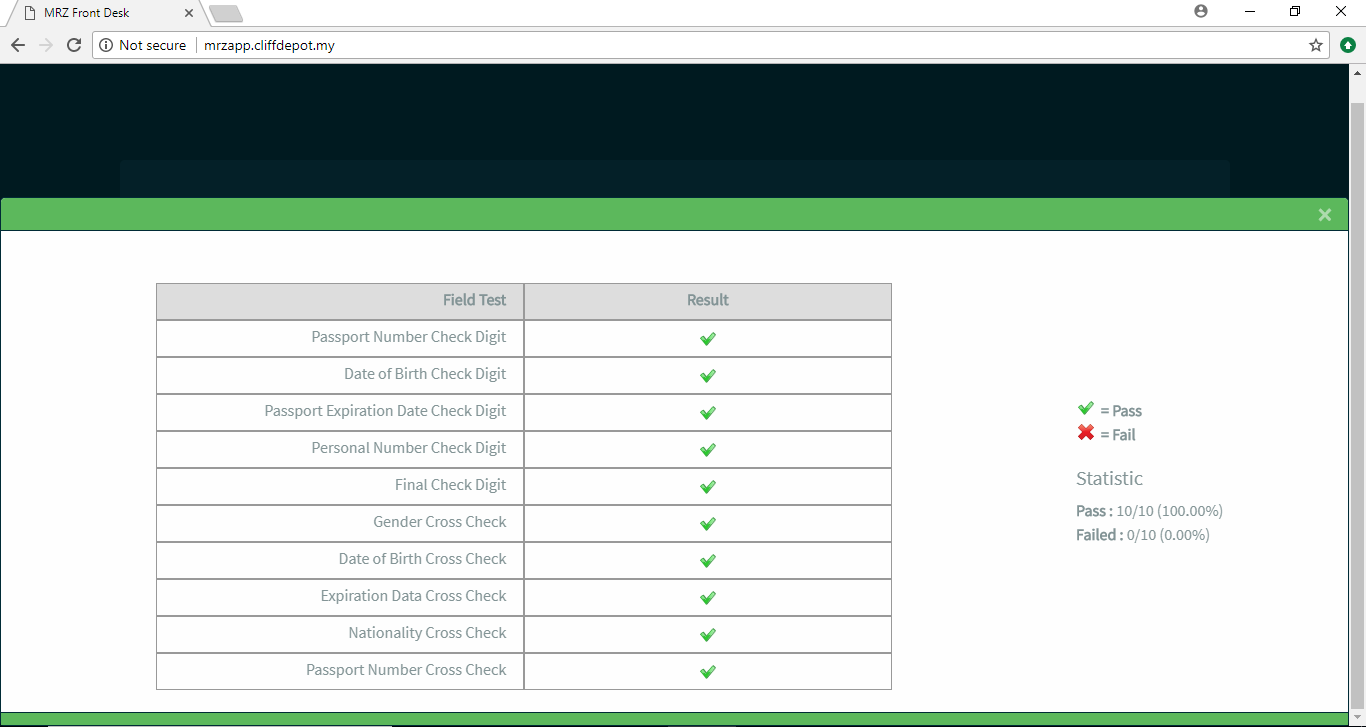
Home Page of MRZ Verifier

The “MRZ (line 2)” is available in two modes – split and normal textbox. The normal textbox is enable user do MZR data paste action when necessary.



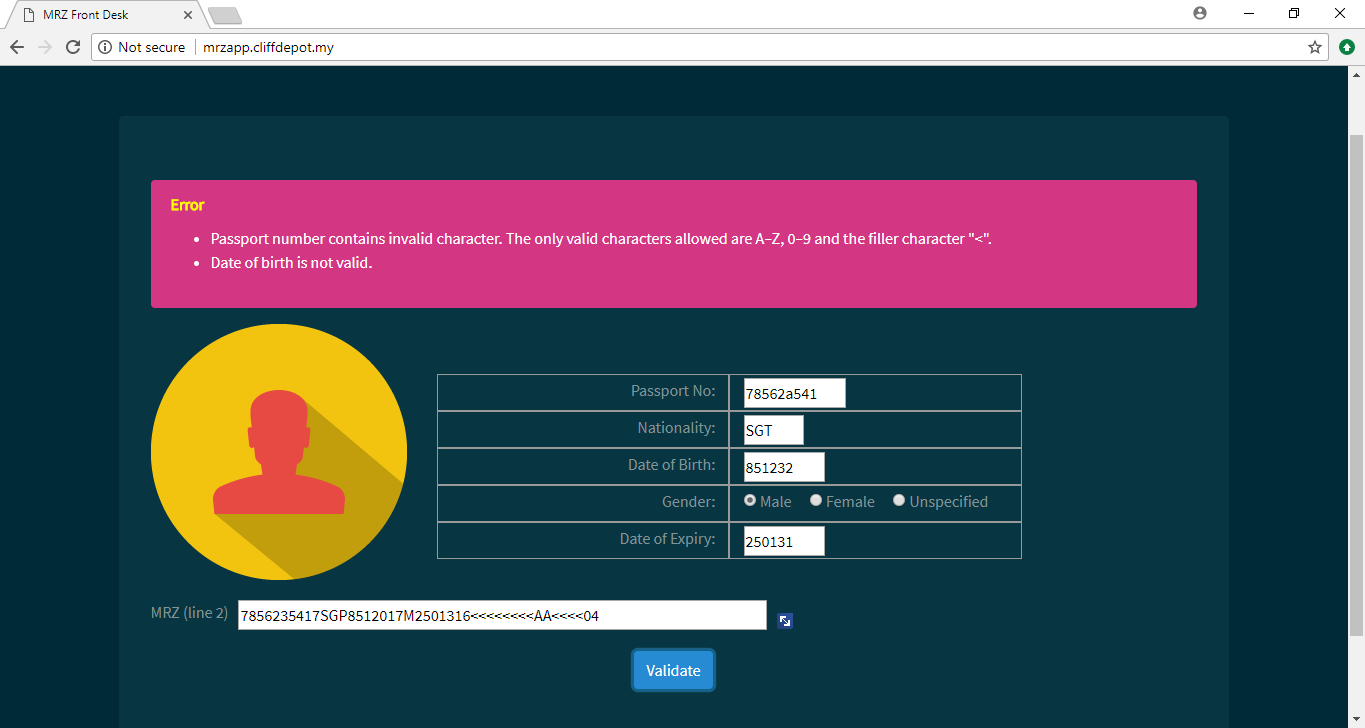
MRZ (line 2) Normal Textbox Mode

Clicking on “Validate” button, result panel is popup. The validation result is indicating each field test outcome and a brief statistic.



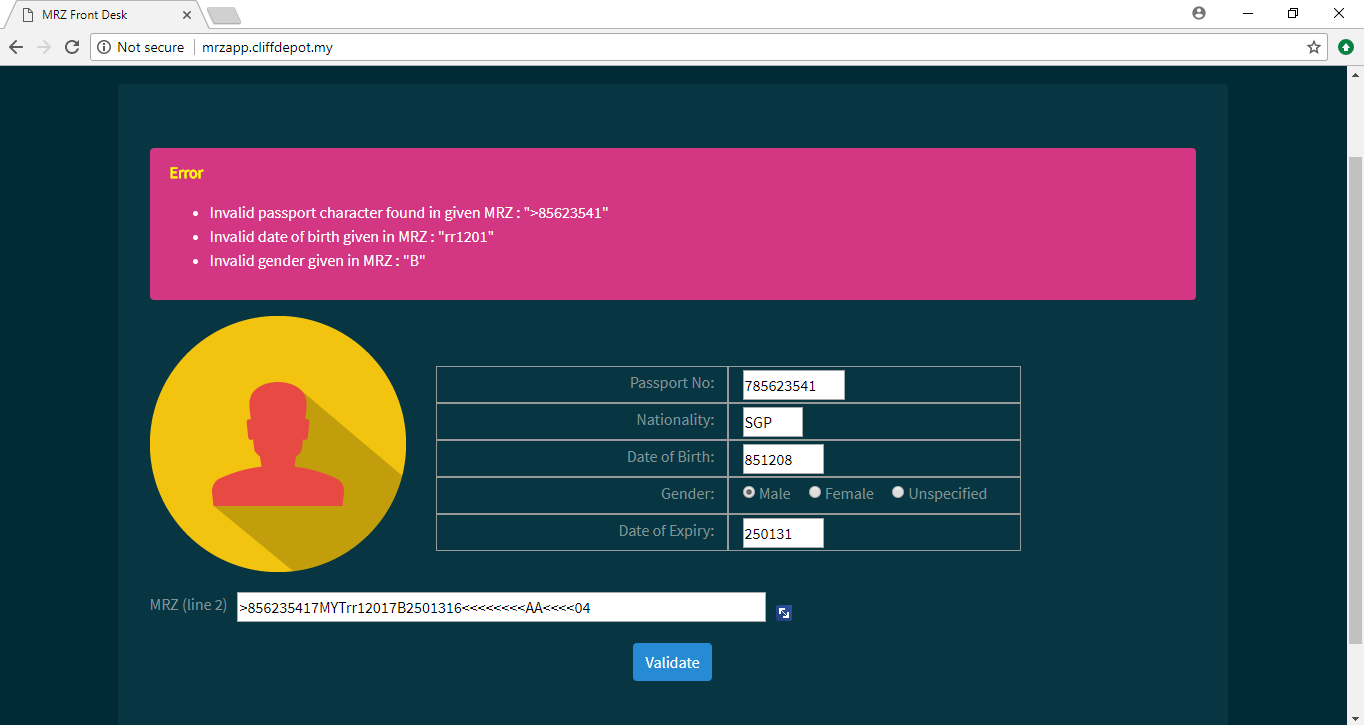
Validation Result Popup

If user provides incorrect input, the frontend validation summary is prompting as below. In general, user needs to provide value for all input form. The input form allows any inputs from user but there is validation process at the end when user clicks on validate button. User needs to note the input value is case sensitive and the date format is compliance to standard “*ddMMyy*”. The data of the “MRZ line 2” consists of 44 characters and the only characters used are *A–Z*, *0–9* and the filler character *<*.



Frontend Validation Summary

Besides, if there is validation error raised at backend WebAPI service, the frontend validation summary is visible as well, for example below:



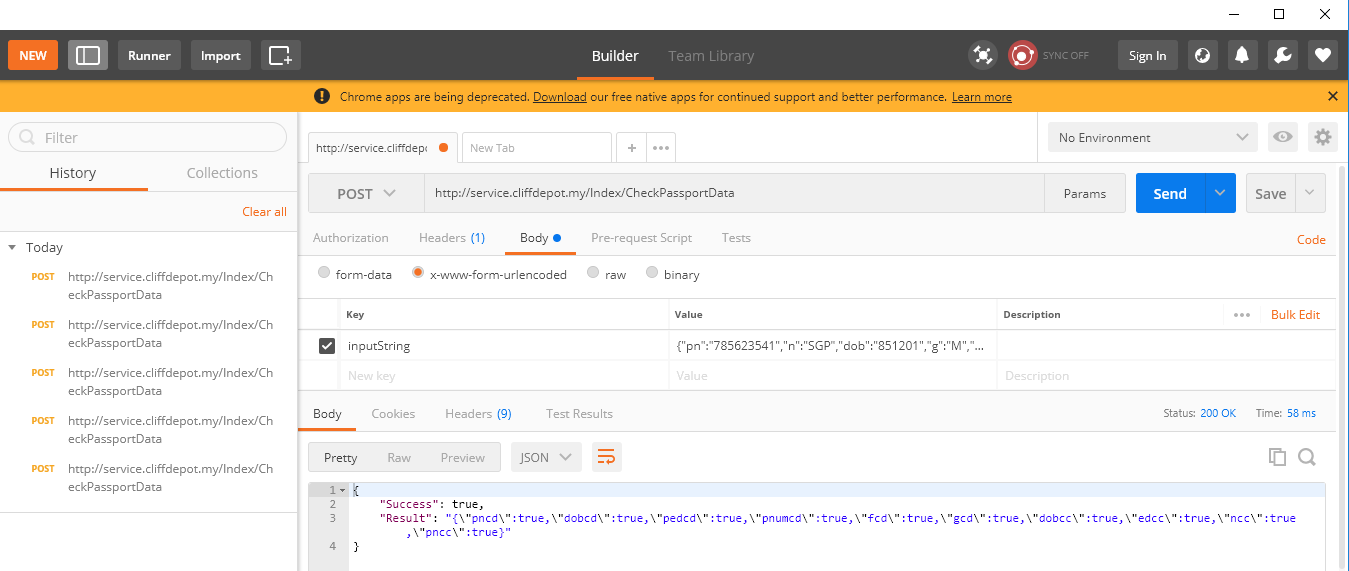
Validation Summary Response from MRZ Verifier API

# Addendum

The below section documented the additional explanation and alternative in testing the MRZVerifier compoenent.

## Invoking Web API Externally via Add-on Tool

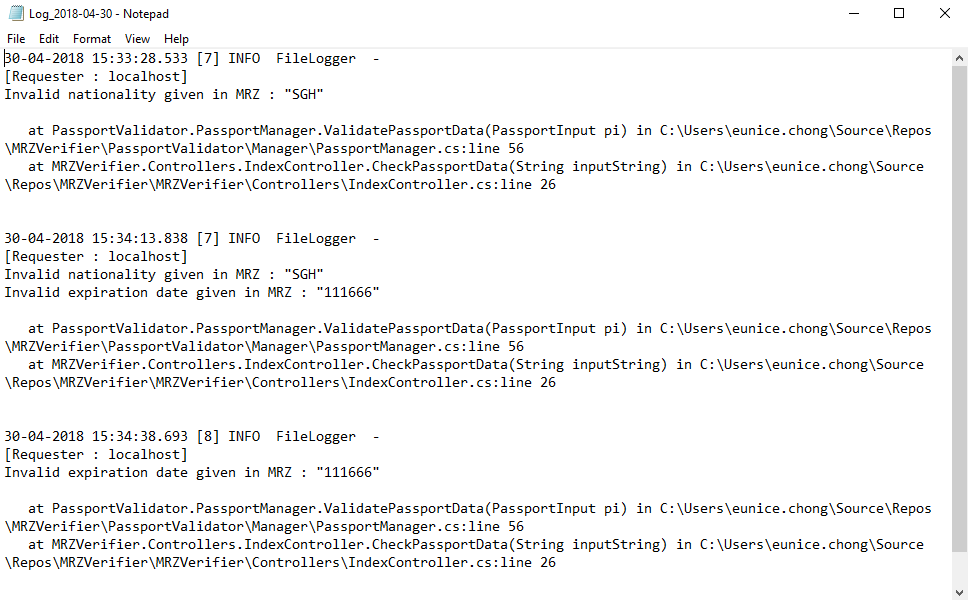
User can utilize any third party tool to invoke the MRZVerifier web API service. For instance, the below figure illustrates Chrome add-on Postman is invoking the web API method, CheckPassportData, with the necessary REST notation data as parameter.



Chrome Postman Add-on

## Log Checking

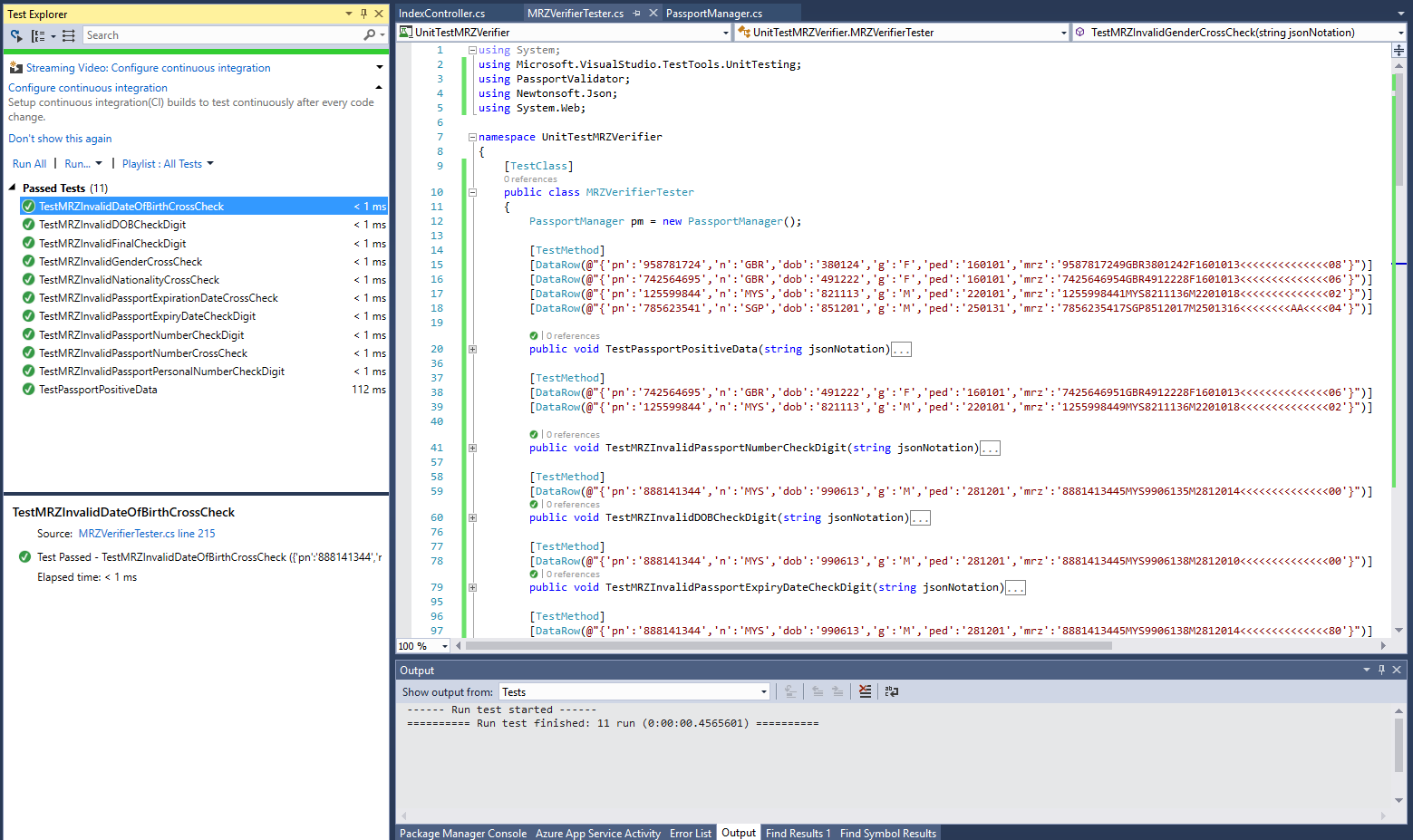
For each user request to MRZVerifier web API, if there is exception throwing, a record is created in a log file with timestamp via log4net. This enables the site administrator to monitor the health of the service apart from tracing the root cause of the error via exception stack detail conveniently. Below figure illustrate a sample log created by the API service in hosting server when user provides invalid input in the frontend.



Sample Log Generated

## MS Unit Test Passport Validator DLL

There is one unit testing project created, namely UnitTestMRZVerifier, which utilizing the Microsoft Test Framework to test the logic of PassportValidator dll. Different unit test case and test method are created to test the validation result outcome. The below figure illustrated the test explorer outcome in MS Visual .NET.



MS Visual Studio .NET Unit Test Explorer

# Link References

<https://en.wikipedia.org/wiki/Machine-readable_passport>

<http://www.highprogrammer.com/alan/numbers/mrp.html>

<https://www.owasp.org/index.php/Top_10-2017_Top_10>

<http://www.emvlab.org/mrz/>