|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | dfwnet | | | | | **Test Case ID** | dfwnet-tc-005 | | | |
| **Test Case Description** | | Each sample downloaded from GitHub compiles and works | | | | | **Test Priority** | High | | | |
| **Pre-Requisite** | | Dotnet core framework installed. Docker service up and running | | | | | **Post-Requisite** |  | | | |
| **Technology Stack** | | .NET Core 2.1 | | | | | **Checked by** | José Manuel Sánchez | | | |
| **Test Procedural Steps** | | | | | | | | | | | |
| **Step** | **Action** | | | **Inputs** | **Expected Output** | **Actual Output** | | | **External Links** | **Remarks** | **Test Result** |
| 1 | AOP.ErrorHandler | | Inside the AOP.ErrorHandler folder the AOP.ErrorHandler solution for .net and .net core.  The project must compile. | | The solution compiles with no error.  The tests must run the show cases to catch exceptions and input params | The solution compiles and runs for both platforms.  The test cases run and show all the inputs. | | |  |  | **OK** |
| 2 | GMailAPIConsumer | | The GMailAPIConsumer must contain the sample for .net and .net core | | The solution for both platforms must compile and run.  The project must run on port 2025.  On the first login the tester must introduce the credentials.  The swagger must run.  The user should be able to send and email this json:  { "EmailFrom":"mythaistarrestaurant@gmail.com", "EmailAndTokenTo":{ "MD5Token1":" Email\_Here!@gmail.com", "MD5Token2":" Email\_Here!@gmail.com" }, "EmailType":0, "DetailMenu":[ "Thai Spicy Basil Fried Rice x2", "Thai green chicken curry x2" ], "BookingDate":"2017-05-31T12:53:39.7864723+02:00", "Assistants":2, "BookingToken":"MD5Booking", "Price":20.0, "ButtonActionList":{ "http://accept.url":"Accept", "http://cancel.url":"Cancel" }, "Host":{ " Email\_Here!@gmail.com":"José Manuel" } } | Both solutions compile and run.  The application is up on the port that it’s specified.  The email is sent with the given json structure. | | | http://localhost:2025/swagger/ | After the first login there’s no need to enter the credentials and give grants again | **OK** |
| 3 | GrayLog | | Inside the GrayLog folder, run the command :  Docker-compose up | | The Docker script runs with no problems. The docker instance starts a Graylog instance at port 9000 | The script runs and start the GrayLog instance at port 9000. The default user is admin admin.  Every time the docker-compose command is run, the data container keeps the information. | | |  | Inside the folder there is the docker-compose.yml file | **OK** |
| 4 | MyThaiStar | | Inside the MyThaiStar folder run the command :  Docker-compose up. | | The script must compile with no errors, create the docker containers and deploy the application.  When the script finishes, on next ports must be deployed the MyThai Star application:  2025: Mail service  4200: Angular Client  8081: .Net core API  2021: Reporting | The Docker-compose up command compiles and deploys the My Thai Star Sample.  The different services runs showing the swagger.  The application My Thai Star runs on port 4200 and is able to show dishes, book tables and send emails. | | | <http://localhost:2025/swagger/>  <http://localhost:4200/swagger/>  <http://localhost:2021/swagger/>  <http://localhost:8081/swagger/> |  | **OK** |

This document is a template for tracking the test case performed for a specific scenario.