Exploring

Self-Sovereign Identity

Testing Standards

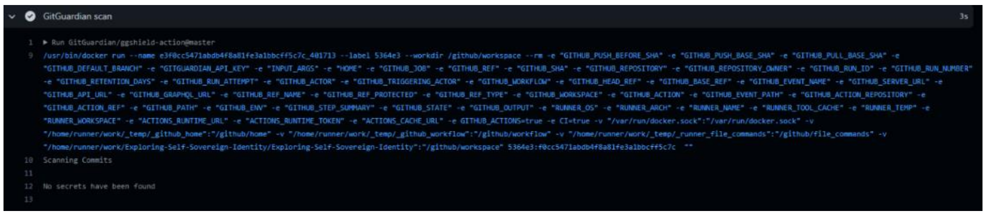
Unit and integration tests were run on all important functional elements of the system. Git Guardian is used for security testing and exposure testing for secrets. To keep the private keys used, secret.

We made use of a singular testing framework from Microsoft MS Test. This was a part of the Visual Studio suite and was chosen for its ease of use and compatibility with the system.

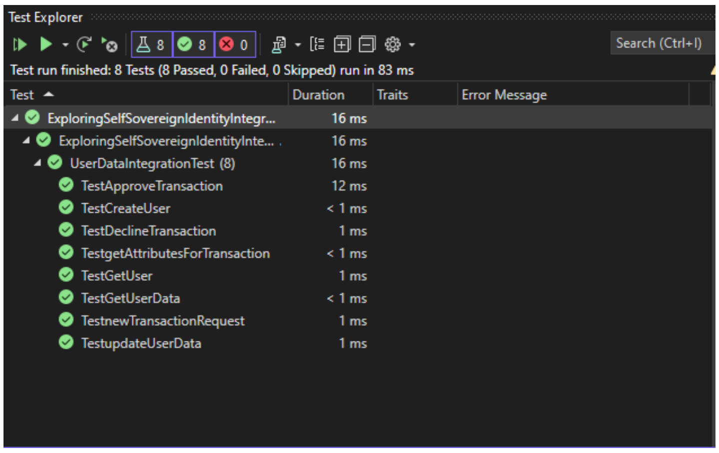
The REST API is tested using that framework.

Integration testing is carried out by injecting a service and using that service to call the functions. We Assert to check if objects are not null and their function response is the same as the expected results.

These tests are integrated with our Github using ci/cd, and it is automated. New code is only merged when the test pass.

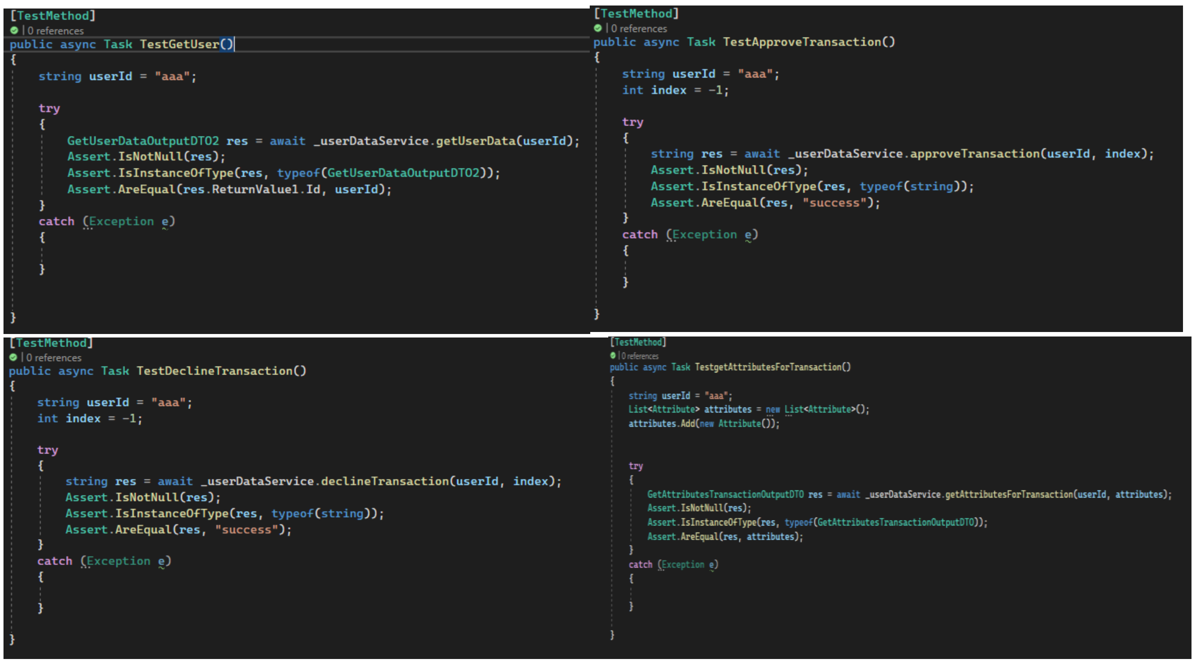


```
GitGuardian scan                                                                                              3s

▶ Run GitGuardian/ggshield-action@master
/usr/bin/docker run --name e3f0cc5471abdb4f8a81fe3a1bbcff5c7c_401713 --label 5364e3 --workdir /github/workspace --rm -e "GITHUB_PUSH_BEFORE_SHA" -e "GITHUB_PUSH_BASE_SHA" -e "GITHUB_PULL_BASE_SHA" -e
"GITHUB_DEFAULT_BRANCH" -e "GITGUARDIAN_API_KEY" -e "INPUT_ARGS" -e "HOME" -e "GITHUB_JOB" -e "GITHUB_REF" -e "GITHUB_SHA" -e "GITHUB_REPOSITORY" -e "GITHUB_REPOSITORY_OWNER" -e "GITHUB_RUN_ID" -e "GITHUB_RUN_NUMBER"
-e "GITHUB_RETENTION_DAYS" -e "GITHUB_RUN_ATTEMPT" -e "GITHUB_ACTOR" -e "GITHUB_TRIGGERING_ACTOR" -e "GITHUB_WORKFLOW" -e "GITHUB_HEAD_REF" -e "GITHUB_BASE_REF" -e "GITHUB_EVENT_NAME" -e "GITHUB_SERVER_URL" -e
"GITHUB_API_URL" -e "GITHUB_GRAPHQL_URL" -e "GITHUB_REF_NAME" -e "GITHUB_REF_PROTECTED" -e "GITHUB_REF_TYPE" -e "GITHUB_WORKSPACE" -e "GITHUB_ACTION" -e "GITHUB_EVENT_PATH" -e "GITHUB_ACTION_REPOSITORY" -e
"GITHUB_ACTION_REF" -e "GITHUB_PATH" -e "GITHUB_ENV" -e "GITHUB_STEP_SUMMARY" -e "GITHUB_STATE" -e "GITHUB_OUTPUT" -e "RUNNER_OS" -e "RUNNER_ARCH" -e "RUNNER_NAME" -e "RUNNER_TOOL_CACHE" -e "RUNNER_TEMP" -e
"RUNNER_WORKSPACE" -e "ACTIONS_RUNTIME_URL" -e "ACTIONS_RUNTIME_TOKEN" -e "ACTIONS_CACHE_URL" -e GITHUB_ACTIONS=true -e CI=true -v "/var/run/docker.sock":"/var/run/docker.sock" -v
"/home/runner/work/_temp/_github_home":"/github/home" -v "/home/runner/work/_temp/_github_workflow":"/github/workflow" -v "/home/runner/work/_temp/_runner_file_commands":"/github/file_commands" -v
"/home/runner/work/Exploring-Self-Sovereign-Identity/Exploring-Self-Sovereign-Identity":"/github/workspace" 5364e3:f0cc5471abdb4f8a81fe3a1bbcff5c7c  ""
Scanning Commits

No secrets have been found

```



```
Test Explorer
8  8  0                                                              Search (Ctrl+I)
Test run finished: 8 Tests (8 Passed, 0 Failed, 0 Skipped) run in 83 ms
Test ▲                                          Duration   Traits   Error Message
◢ ExploringSelfSovereignIdentityIntegr...         16 ms
  ◢ ExploringSelfSovereignIdentityInte...         16 ms
    ◢ UserDataIntegrationTest (8)                 16 ms
        TestApproveTransaction                    12 ms
        TestCreateUser                           < 1 ms
        TestDeclineTransaction                     1 ms
        TestgetAttributesForTransaction          < 1 ms
        TestGetUser                                1 ms
        TestGetUserData                          < 1 ms
        TestnewTransactionRequest                  1 ms
        TestupdateUserData                         1 ms
```

Examples of unit tests:



```
[TestMethod]
0 references
public async Task TestGetUser()
{
    string userId = "aaa";

    try
    {
        GetUserDataOutputDTO2 res = await _userDataService.getUserData(userId);
        Assert.IsNotNull(res);
        Assert.IsInstanceOfType(res, typeof(GetUserDataOutputDTO2));
        Assert.AreEqual(res.ReturnValue1.Id, userId);
    }
    catch (Exception e)
    {

    }

}
```

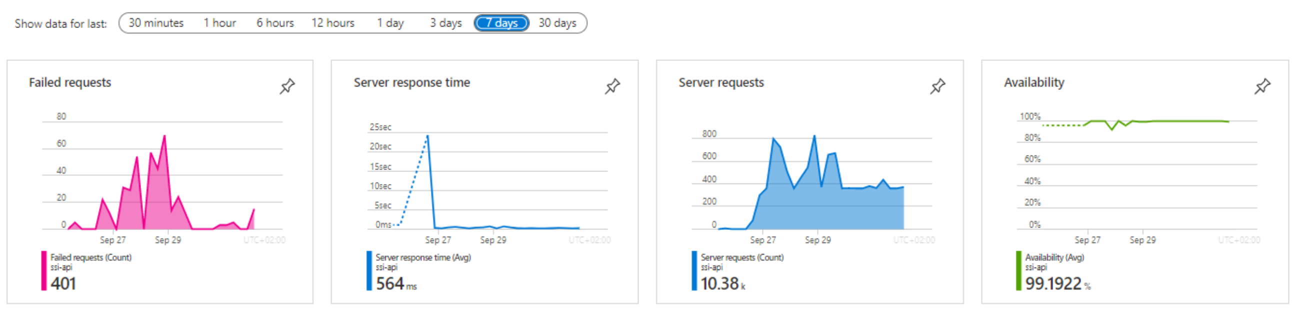
```
[TestMethod]
0 references
public async Task TestApproveTransaction()
{
    string userId = "aaa";
    int index = -1;

    try
    {
        string res = await _userDataService.approveTransaction(userId, index);
        Assert.IsNotNull(res);
        Assert.IsInstanceOfType(res, typeof(string));
        Assert.AreEqual(res, "success");
    }
    catch (Exception e)
    {

    }

}
```

```
[TestMethod]
0 references
public async Task TestDeclineTransaction()
{
    string userId = "aaa";
    int index = -1;

    try
    {
        string res = await _userDataService.declineTransaction(userId, index);
        Assert.IsNotNull(res);
        Assert.IsInstanceOfType(res, typeof(string));
        Assert.AreEqual(res, "success");
    }
    catch (Exception e)
    {

    }

}
```

```
[TestMethod]
0 references
public async Task TestgetAttributesForTransaction()
{
    string userId = "aaa";
    List<Attribute> attributes = new List<Attribute>();
    attributes.Add(new Attribute());


    try
    {
        GetAttributesTransactionOutputDTO res = await _userDataService.getAttributesForTransaction(userId, attributes);
        Assert.IsNotNull(res);
        Assert.IsInstanceOfType(res, typeof(GetAttributesTransactionOutputDTO));
        Assert.AreEqual(res, attributes);
    }
    catch (Exception e)
    {

    }

}
```

Non-functional Requirements testing was done on Azure since the hosting for the system was on Microsoft Azure.

The metrics below are a summary of failed requests, server response time, server requests and availability of the system from [Azure Monitor](#) (Application Insights). Application insights allows us to see all this data in different time frames and inspect each issue or failed request into detail.


Show data for last:
30 minutes
1 hour
6 hours
12 hours
1 day
3 days
7 days
30 days
Failed requests
Sep 27
Sep 29
UTC+02:00
Failed requests (Count)
ssi-api
Server response time
25sec
20sec
15sec
10sec
5sec
0ms
Sep 27
Sep 29
UTC+02:00
Server response time (Avg)
ssi-api
564ms
Server requests
Sep 27
Sep 29
UTC+02:00
Server requests (Count)
ssi-api
10.38k
Availability
Sep 27
Sep 29
UTC+02:00
Availability (Avg)
ssi-api