

Exercise – 2

Test Case – TC002

Functionality – User Login

Environment – Test

Application Url – <https://ensekautomationcandidatetest.azurewebsites.net/>

API - <https://qacandidatetest.ensek.io/ENSEK/login>

Assumptions –

- + API endpoint “/ENSEK/login” is integrated with UI of the application

Details –

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Outcome	Execution Status
TC002	User Login (Valid Credentials)	1. Open the login page 2. Enter valid credentials 3. Click Login button	Email - ***** Password - *****	The user is logged in successfully	FAILED

Reason for failure –

- + User login is happening by giving username and password through API
- + The application is not accepting normal text other than email as username. This contradicts my assumption as part of the integration test
- + Tried with different user credentials, but the application fails every time

Further steps –

- + Need to check with the project manager/ product owner about API and UI integrations from a login perspective
- + If the team accepts on API and UI integration as per my assumption, then I need to question why API performs login with credentials, whereas UI is not accept the same credentials instead asking for a valid email toward username
- + UI is prompting for valid email towards user name. Need to check on use case from business requirements and confirm with product owner/ BA (business analyst)
- + Refer to the server logs to find out the reason for failing to login

API Execution Screenshot –

The screenshot shows an API client interface. At the top, the URL bar displays "https://qacandidatestest.ensek.io/ENSEK/login". The request method is set to "POST". The request body is in JSON format, containing the following data:

```
{  "username": "test",  "password": "testing"}
```

The response status is "200 OK" with a response time of 186 ms and a body size of 577 B. The response body is in JSON format, containing the following data:

```
{  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImh0dCI6MTcyNjcyNTM1MiwiYWVhbnR5IjoiaWw1IiwiaWF0IjE5NS00NTk2LTk4NGQ0tZjBjMDc5YjRmYmI5IiwidHlwZSI6ImFjY2VzcyIsInN1YiI6InRlc3QlLCJyYmYiOiE3MjY3MjUzNTIsImV4cCI6MTcyNjcyNjI1MiwicGFzc3dvcmQ0Ij0ZXN0aW50LmIjwsw9knUzKwkqLyIra-Ehok9TQz3d5J9IwafiXBs",  "message": "Success"}
```

Application Login Screenshot –

The screenshot shows the application login page. The page has a dark header with the "ENSEK" logo and navigation links: "Home", "About", "Contact", "Register", and "Log in". The main content area is titled "Log in." and contains two sections: "Use a local account to log in." and "Use another service to log in." The "Use a local account to log in." section has a form with "Email" and "Password" fields. The "Email" field contains "test@abc.com". The "Password" field is masked with "*****". There is a "Remember me?" checkbox and a "Log in" button. Below the form, there is a link "Register as a new user". The "Use another service to log in." section contains a message: "There are no external authentication services configured. See this article for details on setting up this ASP.NET application to support logging in via external services."

Application response -

The screenshot shows the application response page. The page has a dark header with the "ENSEK" logo and navigation links: "Home", "About", "Contact", "Register", and "Log in". The main content area is titled "Error" and contains a message: "An error occurred while processing your request". Below the message, there is a detailed error description: "A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: SQL Network Interfaces, error: 26 - Error Locating Server/Instance Specified)". Below the error description, there is a stack trace starting with "at System.Data.SqlClient.SqlInternalConnectionTds..ctor(DbConnectionPoolIdentity identity, SqlConnectionString connectionOptions, SqlCredential credential, Object providerInfo, String newPassword, SecureString newSecurePassword, Boolean redirectedUserInstance, SqlConnectionString userConnectionOptions, SessionData reconnectSessionData, DbConnectionPool pool, String accessToken, Boolean applyTransientFaultHandling, SqlAuthenticationProviderManager sqlAuthProviderManager) at System.Data.SqlClient.SqlConnectionFactory.CreateConnection(DbConnectionOptions options, DbConnectionPoolKey poolKey, Object poolGroupProviderInfo, DbConnectionPool pool, DbConnection owningConnection, DbConnectionOptions userOptions) at System.Data.ProviderBase.DbConnectionFactory.CreatePooledConnection(DbConnectionPool pool, DbConnection owningObject, DbConnectionOptions options, DbConnectionPoolKey poolKey, DbConnectionOptions userOptions) at System.Data.ProviderBase.DbConnectionPool.CreateObject(DbConnection owningObject, DbConnectionOptions userOptions, DbConnectionInternal oldConnection) at System.Data.ProviderBase.DbConnectionPool.TryGetConnection(DbConnection owningObject, UInt32 waitForMultipleObjectsTimeout, Boolean allowCreate, Boolean onlyOneCheckConnection, DbConnectionOptions userOptions, DbConnectionInternal& connection) at System.Data.ProviderBase.DbConnectionPool.TryGetConnection(DbConnection owningObject, TaskCompletionSource`1 retry,