Automation Test Assessment Report

# 1. Introduction

This report outlines the implementation of the Automation Test Assessment, assigned as part of the evaluation process for the Senior Automation Test Engineer position . The implementation is based on Robot Framework 4.1.3, Selenium Library 3.41.0 and was created on RIDE editor v2.1b3.

# 2. Implementation Description\*

The solution was designed to fulfill the 5 main tasks described in the assessment:

## 2.1 Task 1 – Filtering Football Matches

A keyword named `Filter Scheduled Matches` was implemented to filter scheduled events and identify only football matches. Locators used for filtering are defined in the `Locators.json` configuration file.

## 2.2 Task 2 – Detecting Match Start Time

A test case tracks the scheduled start time of each match on the 'Live Schedule' page. The script waits for the appropriate time and then checks the 'Live' page to verify whether the event has gone live.

## 2.3 Tasks 3 & 4 – Delayed and Dropped Match Detection

The logic for determining whether a match is delayed or dropped is integrated. If a match does not appear on the live page at the scheduled time, it is marked as delayed. If it still does not appear after 20 minutes, it is considered dropped.

## 2.4 Task 5 – XML Reporting

All delayed and dropped matches are logged in an XML file along with a timestamp (GMT+2) indicating the scheduled live time. This is handled through functionality in the `XML.resource` file.

\*Due to portal dynamic scroller, implementation includes the first 7 matches that locator matches in runtime. Moreover, on the live page due to dynamic scroller we scroll on each card to make visible and loaded each match locator. This is not the optimized way but it could be optimized using all the information about project.

# 3. Extra Points – Justification

The implementation fulfills several criteria for receiving extra points:

• Use of Page Object Model (POM) through dedicated resource files for each portal section and functionality. We have low level resources such as System and Browser, and higher level such as Validators, Reports and PortalPages. Moreover this practice makes Test Cases code stable, only changing what is needed in keywords that are repeatedly used.

• Configuration files Locators.json and Specs.json allows easy maintenance of locators and portal specifications (i.e. change url).

• Suite Setup and Teardown are defined to improve framework stability.

• XML reporting is dynamic and accurately captures delayed and dropped events.

• The entire solution is modular and scalable for future test expansions.

• The entire solution could be used as a framework from scratch.

Note: Localization folder was created in case page texts would be used.

# 4. Conclusion

The developed test framework fully addresses the goals of the assessment. It is cleanly structured, follows test automation best practices, and is built for maintainability and scalability. The solution demonstrates strong technical understanding and readiness for automation tasks in a real-world environment.