IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

APPLICATION OF TEST AUTOMATION FRAMEWORK FOR TESTING OF BIOMEDICAL DESKTOP APPLICATIONS

[1]Lathangi N, [2]Dr. Prasanna Kumar S.C

[1]Final Year Student, [2]Professor

[1]Department of Electronics And Instrumentation Engineering,

[1]RV College of Engineering, Bengaluru, India

Abstract: The intricate nature of healthcare applications, handling sensitive patient data and influencing critical medical decisions, demands rigorous testing strategies. While manual testing remains vital, its limitations in efficiency and error susceptibility hinder comprehensive software evaluation. Test Automation Frameworks (TAFs) emerge as a powerful solution, streamlining the testing process for healthcare applications. We explore how TAFs enhance testing efficiency by automating repetitive tasks, allowing testers to focus on high-level and exploratory scenarios. This not only reduces testing costs but also leads to improved test accuracy and repeatability. TAFs enable running a broader spectrum of test cases, encompassing regression testing, edge cases, and integration testing, resulting in more comprehensive test coverage and reduced risks of undetected defects. This paper delves into the multifaceted advantages and practical uses of TAFs within the healthcare software domain. This paper further explores potential future directions for TAFs in the healthcare domain. Advancements in machine learning and artificial intelligence hold promise for enhanced test automation capabilities, enabling automated test generation and self-healing test scripts. Integrating TAFs with continuous integration/continuous delivery (CI/CD) pipelines streamlines the development process for healthcare applications. Embracing TAFs and staying at the forefront of technological advancements will allow developers and healthcare institutions to guarantee the quality and reliability of healthcare applications, ultimately contributing to improved patient care outcomes.

Index Terms - Software Testing, Test Automation Frameworks (TAFs), Page Object Model, Healthcare Quality Management.

I. INTRODUCTION

Healthcare applications play a vital role in modern medical care, managing patient records, facilitating diagnoses, and supporting treatment decisions. Ensuring the reliability and functionality of these applications is paramount to patient safety and healthcare delivery efficiency. Traditional manual testing methods, while crucial, struggle to keep pace with the rapid development cycles and growing complexity of healthcare software The paper delves into specific uses of TAFs within healthcare applications. This includes automating tests for core functionalities like patient data management, medication administration, and order entry. Regression testing becomes more efficient by automatically re-running existing test cases, ensuring new versions maintain functionality. Non-functional aspects like performance, security, and usability within healthcare applications can also be automated. Verifying seamless data exchange and interoperability between healthcare applications and other systems through TAFs promotes robust integration. TAFs facilitate faster feedback loops through quicker and more frequent test execution, enabling software development teams to identify and rectify bugs promptly, leading to faster development cycles and improved software quality. Lastly, automated API testing streamlines the communication verification between healthcare applications and external systems.

However, implementing TAFs necessitates careful consideration. Initial investments in infrastructure, tools, and expertise are required. Maintaining test scripts as healthcare applications evolve is crucial for their continued effectiveness. Selecting the most suitable TAF for each application demands a thorough evaluation of its functionalities and complexity.

II. HIGH LEVEL ARCHITECTURE

This paper demonstrates the attributes and functioning of the TAF using a Windows Desktop Calculator as the Application Under Test. A robust test automation framework is crucial for ensuring the quality and functionality of healthcare software. Such a framework would provide a structured approach to design, develop, and execute automated test cases. Ideally, it would be scalable and adaptable to accommodate the diverse functionalities of various healthcare applications. This could involve supporting multiple scripting languages and integrating with industry-standard testing tools. A key aspect would be the ability to manage test data effectively, considering the sensitive nature of healthcare information. By incorporating data-driven testing practices, the framework could handle a wider range of test scenarios with dynamic data sets. Ultimately, a well-designed test automation framework for healthcare software would streamline the testing process, improve efficiency, and contribute to the delivery of high-quality healthcare IT solutions.

Components

This robust framework is developed using C# (OOP programming language) and is used for automation testing of desktop applications. These frameworks offer flexibility and support for various scripting languages like Python or Java. The attributes/building blocks of the TAF are as follows:

1. Test Suite Design: The framework would house a comprehensive test suite encompassing individual test cases for each software function.
2. Data Logging: Test cases would simulate data recording by sending pre-defined data sets and demographics. The framework would then verify if the software accurately logs and stores the information.
3. Data Reports: Tests would generate sample reports and validate their content against expected formats. This could involve checking for inclusion of patient information, all medical and examination data summaries, and interpretation results.
4. Object Repository: Any Test Engineer who wishes to deploy the TAF to test his AUT must compulsorily create an Object Repository consisting of a map of all his UI/UX controls and must specify his data arguments such as those for Text box, list Items etc.

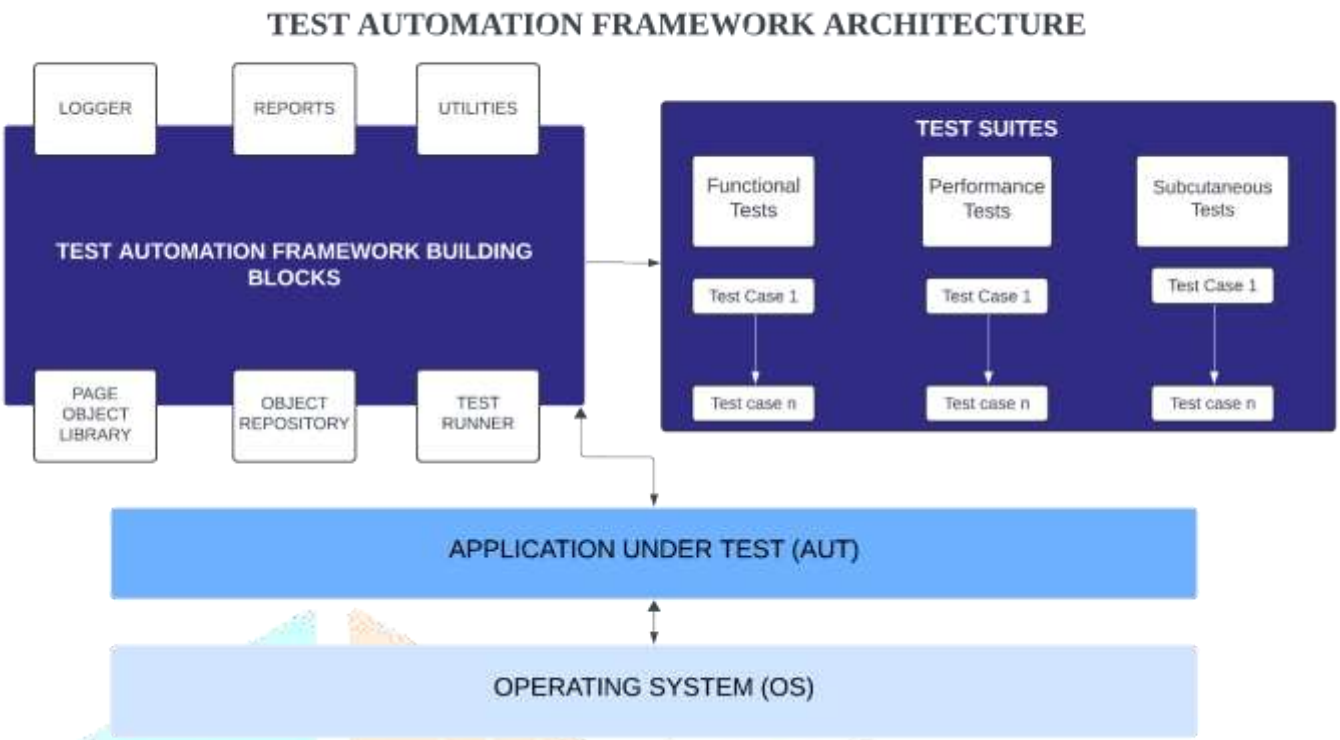

Figure 1: High Level Architecture of Test Automation Framework

III. METHODOLOGY OF AUTOMATION TESTING

We follow these steps in order to deploy the framework to Test automate our AUT:

1. Identify the AUT and It's attributes: The very first step of Test Automation is to define which application we need to test and the technology it is based off of, the platform of the application i.e. Desktop application or web application or mobile application etc. The TAF and It's functionality is chosen purely based on the type of AUT the tester wishes to test. Our proposed TAF has been created using C# programming language. Install the chosen framework and any additional required libraries or tools (e.g., programming languages, IDEs) and Configure the framework for the specific desktop application technology (e.g., Windows Forms, Java Swing).

2. Test Object Identification: Define a clear approach for identifying UI elements within the application. This could involve

using:

- Object Repository: A centralized location to store locators (identifiers) for UI elements like buttons, text fields, etc. This ensures test stability even with minor UI changes.
- Page Object Model (POM): Creating separate classes representing application pages, encapsulating UI element interaction methods. This improves code organization and maintainability.

3. Test Case Design: To design a sequential Test script for each functionality.

- Develop a comprehensive test suite covering various functionalities of the AUT.
- Prioritize critical user journeys and functionalities for initial automation.
- Design modular test cases with clear steps, expected results, and error handling mechanisms.
- Leverage data-driven testing by using external data files for test inputs and expected outputs.

4. Script Development:

- Write test scripts in the chosen framework's scripting language (e.g., Python, Java, C#).
- Utilize framework functionalities for interacting with UI elements, simulating user actions, and
- asserting expected outcomes.
- Implement proper logging and reporting mechanisms to track test execution status and generate
- detailed reports.

5. Integration and Execution:

- Integrate all test scripts within the framework's structure.
- Develop a mechanism to trigger test execution (e.g., command line, script).
- Schedule automated test runs periodically or as part of the development cycle.

6. Maintenance and Improvement:

- Regularly update the test suite as the AUT evolves.
- Address new functionalities and bug fixes in the application.
- Review and improve test scripts for efficiency and maintainability.
- Monitor test results and identify areas for further automation or improvement.

IV. THE AUT- BIOMEDICAL DESKTOP APPLICATIONS

Most Biomedical Applications are developed to act as a Gateway/Interface Between the Device and a computer or a Patient Monitor where the data is displayed. It can also convert raw data from the device to various formats such as DICOM (Digital Imaging and Communications in Medicine) that are approved for Transmission and storage. The other important use case of these Applications include creating 'orders' for the device or for managing patient Directories which stores the Basic Information of a patient such as Name, Age, Gender and other details which maybe essential for proper diagnosis. One Example would be an application used for ECG management systems designed for providing the necessary tools to measure, review and export compliant and annotated ECGs.

ECG data can be exported in XML, PDF and TIFF formats. Customizable automatic processing rules route ECGs to any destination, for any reason. ECGs can be printed, emailed, faxed, exported and added to worklists based on ECG status, demographics, automatic measurements, interpretation and acquisition priority flag. Configurable worklists organize ECGs according to a study's workflow processes. Worklists give quick access to ECGs requiring demographics verification, measurements and quality checks.

V. BENEFITS OF TEST AUTOMATION FOR BIOMEDICAL APPLICATIONS

The complexity of biomedical desktop applications needs robust testing strategies. Traditional manual testing techniques are often inadequate in terms of efficiency and comprehensiveness. Test automation frameworks offer a solution by automating repetitive test cases, allowing for faster feedback cycles and improved software quality.

- Increased Efficiency: Automating repetitive tasks frees up valuable time for testers to focus on exploratory and complex testing scenarios.
- Improved Accuracy and Repeatability: Automated tests are less susceptible to human error, ensuring consistent and reliable test execution.
- Faster Feedback Cycles: Automating test suites enables faster regression testing after code changes, accelerating development cycles.
- Enhanced Coverage: Automated tests can cover a broader range of test cases compared to manual testing, leading to more comprehensive test coverage.

VI. CHALLENGES OF AUTOMATING BIOMEDICAL DESKTOP APPLICATIONS

1. Regulatory Requirements: Regulatory compliance in the medical field demands rigorous and confidential testing processes as well-documented and auditable results**.** Thus, it is important to note that data of the AUT as well as the TAF functionalities are well equipped with data protection.
2. GUI Complexity: Biomedical applications often have complex user interfaces with non-standard controls, making them challenging to automate using traditional web testing tools.
3. Data Sensitivity: Biomedical data requires special handling and security considerations during automated testing since sensitive data such as patient data may be required as model data set. We must ensure no data breach occurs at any point of time during testing and compilation of test report.

VII. CHALLENGES FACED DURING DEVELOPMENT OF TEST AUTOMATION FRAMEWORK

1. Choosing the Right Tools and Technologies as well as Tool Compatibility
   Ensuring compatibility between the chosen framework, desktop automation tools (like Robot Framework, Appium Desktop), and the target application under test can be tricky.

2. Handling UI Complexity
   - Non-Standard Controls: Biomedical and other desktop applications often have user interfaces (UIs) with non-standard controls that traditional web testing tools might not handle effectively.
   - Frequent UI Changes: Applications are subject to UI updates, and the TAF needs to be adaptable to such changes to maintain test suite validity.

3. Data Management
   - Data Sensitivity: Biomedical applications often deal with highly sensitive patient data. The TAF needs to ensure secure handling of such data during testing, potentially requiring anonymization techniques.
   - Data Variability: Creating and managing diverse test data sets that reflect real-world scenarios can be a complex task.

4. Maintenance and Scalability
   - Maintaining the Framework: As the application under test evolves, the TAF needs to be updated to reflect these changes. This ongoing maintenance effort requires resources and planning.
   - Scalability for Large Test Suites: As the number of test cases grows, the TAF needs to be scalable to handle a larger test suite efficiently without performance degradation.

5. Integration Challenges
   - CI/CD Integration: Integrating the TAF with continuous integration/continuous delivery (CI/CD) pipelines for automated testing as part of the development process can be complex and require additional configuration.
   - Third-Party System Integration: If the application interacts with external systems, the TAF needs to be designed to handle these integrations effectively during testing.

6. Resource Constraints
   - Technical Expertise: Implementing and maintaining a robust TAF often requires skilled personnel with expertise in automation tools, testing methodologies, and programming languages.
   - Time and Budget: Developing and deploying a TAF can be time-consuming and requires a dedicated budget for resources and tools.

VIII. CONCLUSIONS AND FUTURE SCOPE

This research paper provides a foundational framework for developing and applying test automation for biomedical desktop applications. By leveraging the benefits of automation, developers and testers can ensure the quality and reliability of these critical software tools used in healthcare settings.

The Future Scope of this project would be to customized test automation framework specifically tailored for biomedical desktop applications offering significant rise in test speed and efficiency benefits. By addressing the unique challenges of this domain, the proposed framework can improve accuracy and compliance,

ultimately leading to the development of more reliable and robust biomedical software. It is crucial to emphasize that although a Test Automation Framework cannot eliminate manual testing by a Hundred Percent, it can certainly reduce the workload of a tester, while also providing robust and Time-Efficient Solutions to a Company that is developing Biomedical softwares or has been assigned to test the same.

IX. ABBRIVIATIONS

1. API- Application Programming Interface
2. AUT- Application Under Test
3. CI/CD- Continuous Integration/Continuous Delivery
4. DICOM- Digital Imaging and Communications in Medicine
5. ECG- Electrocardiogram
6. GUI- Graphical User Interface
7. TAF – Test Automation Framework
8. UI/UX- User Interface/User Experience
9. XML- Extended Markup Language

REFERENCES

[1] Vega, D. E., Schieferdecker, I., & Din, G. (2010). Design of a test framework for automated interoperability testing of healthcare information systems. In 2010 Second International Conference on eHealth, Telemedicine, and Social Medicine (pp. 1-6). IEEE.

[2] Gupta, G., et al. (2023). Usefulness of exercise stress test in early diagnosis of coronary artery disease in diabetic patients. Journal of Medical Society, 37(1), 20-25..

[3] Resnekov, L. (1971). Automation in cardiology. Heart, 33(Suppl), 194-202.

[4] Carvalho, A. K., Silva, A. G., & Garcia, A. C. B. (2016). Test automation framework for healthcare information systems using a model-based approach. Journal of Systems and Software, 117, 341-354.

[5] Al-Daccak, W., & Abusorrah, A. M. (2019). A novel framework for automating regression testing of medical imaging software. Journal of Medical Imaging and Health Informatics, 10(2), 1126-1133.

[6] Chowdhury, S., & Chandra, S. (2018). A framework for automated testing of clinical decision support systems. Journal of the American Medical Informatics Association, 25(2), 232-239