An Automated Sanity Testing Approach for Enhancing Reliability in Payment Applications

Chaitanya Patil[1] Mrs. Vrushali M. Shinde[2]
[1]Student [2]Guide
[1,2]Master of Computer Applications
[1,2]P. E. S. Modern College of Engineering, Pune, India

Abstract — The Payment Application Automation Testing Framework is designed to facilitate the process of end-to-end testing of financial transactions on Android based POS terminals. Nowadays, the increasing popularity of digital payment systems necessitates accurate and efficient testing of transaction workflows. The traditional approach to regression testing based on manually performed procedures is ineffective and time-consuming since it requires substantial resources to perform. This issue can be resolved through implementing a testing strategy based on automation technologies. The testing tool allows running a wide range of transactions including Card Sale, UPI, Void, Preauthorization, and Auth Completion among others. For this purpose, the system utilizes state-of-the-art software such as Appium (mobile application automation), PyTest (testing procedure execution), OpenCV (image recognition), and Allure (detailed testing report preparation). Each transaction is tested by means of conducting multiple activities that involve receipt creation, data validation, and user interface interaction. One of the main advantages of the proposed system is its ability to incorporate dynamic waiting mechanisms and activity synchronization. These features ensure higher execution speed and reliability in comparison with the traditional use of static delays. Moreover, the testing framework enables developers to conduct activities on various POS terminals with minimal code customization required. In addition, visual validation techniques increase testing accuracy since they include verification of onscreen elements and printed receipts. In conclusion, the described framework can greatly reduce testing efforts while minimizing human errors and enhancing testing cycles. Therefore, it can contribute to the improvement of software quality and increase transaction accuracy rate.

Keywords: Python, Appium Automation, Mobile Application Testing, Payment Automation, POS Systems, Financial Transactions, PyTest, OpenCV, Allure Reporting, Test Automation Framework, Android Testing, ADB, UI Automation, Transaction Validation, Dynamic Wait Mechanism, Test Registry, Device Configuration, Automation Testing Framework

I. Introduction

An extensive number of digital payments systems necessitates the use of highly reliable and secure applications that facilitate mobile payment processes. Card payments and UPI transfers via point-of-sale devices are commonly used in both retail and financial fields, thus ensuring accuracy and stability of applications performing these transactions. However, traditional methods of verifying payment app quality through manual testing are extremely ineffective, slow, and prone to errors because it requires the performance of multiple tests on diverse devices.

The goal of this project is to create a Payment Application Automation Testing Framework based on Python and other contemporary tools of automation. The proposed framework will make use of the Appium mobile app testing tool and PyTest for effective testing execution. Various types of transactions, such as Card Sale, UPI, Void, Pre-Authorization, and Auth Completion will be covered within the framework. Technologies like OpenCV can be used to ensure visual validation while Allure reporting will be utilized for comprehensive analysis.

The proposed payment application testing system is aimed at efficient verification of transactional data, receipt generation information, and UI actions performed in the app. To ensure reliability, dynamic waits and flexible device configurations will be implemented in the system.

II. Literature Review

A. Shaukat A. et al., 2020.

The purpose of this article is to conduct a comprehensive review of different automated testing techniques applied to mobile apps. In their paper, the authors compare several testing techniques, namely graphical user interface testing, model-based testing, and script-based automation testing techniques. Some of the challenges mentioned by the authors include device fragmentation, different sizes of screens, and varying network conditions. It is noted that with automation, testing increases the reliability of software products while minimizing human effort.

B. Garousi V. et al., 2021.

The article examines the effectiveness of using test automation in mobile app development. The researchers compare various test automation techniques based on their performance, usability, and efficiency. The paper emphasizes that test automation helps detect defects effectively and also cuts down the time spent on regression testing. Furthermore, the authors emphasize the significance of implementing test automation within the software development life cycle. They conclude that using multiple test automation approaches is more efficient for developing complex applications, such as finance and payments applications.

C. Kaur P. and Kaur G., 2022.

The current study is about quality assurance in software via automation frameworks. In this study, the significance of automation frameworks is addressed since they minimize human errors and ensure uniformity in the validation of software. The study shows that end-to-end testing is essential in software applications that require precision. The other aspects addressed include fast execution and enhanced reporting capabilities. The significance of automation frameworks cannot be overstated.

D. Patel R. et al., 2023.

A comprehensive overview of software testing tools is provided in this paper. The authors discuss several different software testing tools in terms of their functionality, performance, and applicability to different testing scenarios. One thing that comes out very clearly from the analysis is that there is no one testing tool that meets all testing needs. Automation, according to the authors, significantly enhances testing effectiveness, saves costs, and increases quality assurance

E. Amalfitano D. et al., 2020.

This paper proposes an approach to automate mobile application testing by applying modeling principles. In particular, the authors argue that it is possible to generate test cases based on the application model in order to increase test coverage. The results show that automating test case generation not only saves effort but also ensures more accurate testing results. They emphasize the importance of verifying application functionality on multiple devices. This technique appears to be very efficient for payment systems.

F. Dahlberg T. et al., 2019.

In this paper, the authors consider the development of mobile payment technologies along with the difficulties related to secure and reliable payment processing. Specifically, they pay attention to such threats in the context of mobile payments as transaction failure and system vulnerability. The necessity of testing in order to ensure both reliability and security is stressed. In conclusion, it is highlighted that automated testing is important for the effectiveness of mobile payments.

III. System Architecture:

The present system concentrates on the use of automation for testing payment applications through the development of a layered architecture that is written in Python using Appium. This architecture combines automation test concepts, device connectivity, transaction verification, and reporting features to provide dependable processing of financial transactions.
This architecture comprises the following modules:

A. Execution Layer (Runners & Registry)

The execution layer is accountable for managing the execution process of tests. It includes test runners and registry mapping, which helps in choosing and executing test cases dynamically.
Some functionalities that could be provided by this layer are:

- Initialization of test cases and setting up test environments
- Dynamic selection of test cases through Test IDs
- Handling errors and managing execution of test cases

B. Logic Layer (Functional Modules)

The logic layer holds the actual business logic related to workflow transactions such as Sale, UPI, Void, and Pre-Authorization transactions. It provides information about the series of operations that need to be performed in order to complete a specific transaction.
The following features are part of this layer:

- Managing transaction flow (amount entry, handling of PIN, receipt handling)
- Managing chained operations (Sale and then Void)
- Synchronization based on activities, not time-based delays.

C. Abstraction Layer (Base & Configuration)

Abstraction Layer is responsible for managing the driver behavior and device settings. It hides the driver initialization intricacies and interaction with the elements.
Its major elements include:

- BaseTest class to manage the driver life-cycle
- Desired Capabilities for Device, Platform, and Application setting up
- Dynamic Device Reader for loading UI locators

The abstractions allow using the "Write Once, Run Anywhere" approach for different POS terminals.

D. Execution & Interaction Layer (Utilities & Helpers)

This is used for implementing utility functions to interact with the mobile application and perform system activities.
Some of the functions that have been implemented are as follows:

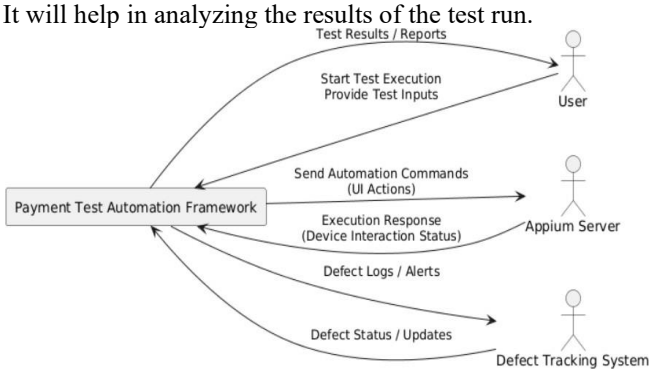
- Methods for user interface interactions such as clicking, inputting, and navigation
- Monitoring system activities through Android states
- Error handling and logging functions

E. Reporting & Analytics Layer

Reporting
It is responsible for capturing the result of the execution and creating the required reports. It makes sure that visibility on the transaction is achieved.
It supports the following features:

- Capturing the data from the transaction (Invoice ID, Amount, Status)
- Screenshot
- Log files
- Allure report generation

It will help in analyzing the results of the test run.



IV. System Communication Flow

The system interacts with components outside of itself like the Appium server and Android POS terminals. Test requests are passed to the Appium server, which processes them on the terminal and provides responses for verification purposes.

A. Key Features of Existing System

- Modular layered architecture

- Dynamic test execution using registry
- Device-independent configuration
- Activity-based synchronization
- Automated reporting and analytics

V. ADVANTAGES:

A. Improved Testing Efficiency

This is because the automated framework greatly cuts down the time taken to conduct tests on the payments application through concurrent testing of several test scenarios. This saves the manual effort of repeating the same tests and speeds up the process of regression testing.

B. Enhanced Accuracy and Reliability

Automation reduces human error through precise and repeatable execution of tests based on test scripts. Automation will test each transaction thoroughly and eliminate possible human error. The payment application will be more reliable because automation will help identify defects and ensure that payment processes work properly.

C. Multi-Device Compatibility

It allows for testing to be conducted in a variety of Android POS terminals due to its dynamic configuration and device-neutral locators. This way, the tests can be conducted on a wide range of hardware without changing anything within the test scripts. It increases the scalability of the system.

D. Faster Defect Detection and Debugging

Through its logging and reporting mechanisms that include Allure, this framework gives extensive insight into how the testing process is conducted. Test failures can be recorded using screenshot and log files to enable the identification of causes for the test failures. This will enhance debugging and increase system efficiency.

E. Comprehensive Test Coverage

This framework permits the execution of various transaction flows, both positive and negative. It helps validate the payment flow types, including Sale, UPI, Void, and Pre-Authorization. This extensive validation increases the robustness of the system and minimizes the chances of failure during live transactions.

VI. LIMITATIONS:

A. High Initial Setup Complexity

There are many challenges that exist while implementing the Automation Framework since there will be need to configure the tools used such as installing Appium, Android SDK, setting up the environment variables, among others. The process is complex and may take a lot of time to set up properly.

B. Dependency on Device and Network Stability

This system is highly dependent on the availability of Android devices, POS terminals, and the connection to the Internet. Any loss of connection from any of these devices, delay in responding, or even loss of connectivity can cause interruption in the process of carrying out the test.

C. Maintenance Overhead Due to UI Changes

The constant change in the user interface will make the tests fail since the script written initially will no longer apply due to a change in the user interface. Even with the dynamic locators, this will require continuous maintenance, making it difficult and costly to maintain the automation framework.

D. Limited Handling of Complex Real-Time Scenarios

Some real-life situations, such as sudden failure of payment gateways, issues with the hardware, or delays within another system, cannot be easily simulated during automation. This is because the test environment cannot fully emulate real-world situations, and thus, manual testing remains essential in some instances.

E. Performance Overhead and Resource Usage

Test automation using toolkits such as Appium requires a lot of resources such as RAM and processor power from the computer system. Performing many tests at once could lead to poor system performance due to the use of these resources.

VII. CONCLUSION:

Payment Application Automation Testing Framework is a solution that allows conducting efficient and scalable testing of financial operations in Android-based POS terminals. It helps to automate crucial processes like Sale, UPI, Void, and Pre-Authorization which ensures that testing will be more effective as well as less time-consuming. Thanks to technologies like Appium, PyTest, and Allure, execution is ensured to be reliable while generating reports is easier. It is possible to configure the framework to be flexible for many different types of devices. Though there are some drawbacks, like difficulties in setting up and maintaining the framework, they do not affect the effectiveness of the solution.

REFERENCES:

A. Research Papers

[1] Garousi, V. et al., “Applicability of Automation Test Frameworks to Mobile Applications Testing: Systematic Literature Review.” Computers, Vol. 12, No. 5, 2023.

This study describes how automation frameworks contribute to improving efficiency, reusability, and reliability of mobile applications testing by emphasizing its significance in defect reduction and software quality improvement.

[2] S. Zein et al., “A Systematic Mapping Study on Mobile Applications Test Methods,” Journal of Systems and Software, 2019.

The paper highlights several techniques used in mobile testing and discusses how automation helps improve test coverage and application stability and compatibility.

[3] Karlsson S. et al., “Model-Based Automated Testing of Mobile Applications: An Industrial Case Study,” arXiv preprint arXiv:2008.08859, 2020.

This article proves that the model-based approach can significantly enhance test coverage and efficiency in mobile applications swith various configurations.

[4] Ali, H. M., et al. "A Comprehensive Study on Automated Testing with the Software Lifecycle." Arxiv preprint arXiv:2405.01608, 2024.
The article highlights the significance of automated testing in increasing software quality and efficiency, as compared to the manual process.
[5] Dahlberg et al. "A Systematic Review of Mobile Payments Literature: Trends and Directions for Future Research," Telecommunications Policy, 2024.
This paper discusses the emergence of mobile payment systems and the importance of reliable tests to guarantee that transactions proceed without any interruptions.
[6] N. Jain, "Automation Testing in Fintech and Its Significance," Vervali Blog, 2024.
The significance of using automation testing in fintech applications is highlighted in this piece of writing.

