International Journal of Multidisciplinary Research and Growth Evaluation.

Enhancing UAT Precision for Cloud-Based Billing apps Using Json Payload Validation in Postman

Udayan Verma
Denver, USA

* Corresponding Author: **Udayan Verma**

Article Info

ISSN (Online): 2582-7138
Impact Factor (RSIF): 8.04
Volume: 03
Issue: 06
November- December 2022
Received: 06-10-2022
Accepted: 09-11-2022
Page No: 997-1000

Abstract
User Acceptance Testing (UAT) is a significant component of cloud billing system. It does this by ensuring that in a hybrid Azure environment, the system functions properly, that financial calculations are correct and that it adheres to all rules prior to launching. This paper discusses the use of Postman to automate tests and also reasons why the use of the JSON payload format should be adopted to raise the accuracy of UAT. Acceptance of complex API responses reduces human errors and shortens billing data, tax evaluation and adjustment approval. The method ensures that there is a strong connection between the back-end logic and the business expectations and give the stakeholders and QA engineers a chance to work effectively together in the process of validation. Evaluations indicate that the speed, precision and reliability of the billing procedure have greatly increased. The results show that UAT based on automated JSON validation not only minimizes the time of testing but also increases the reliability and the credibility of cloud-based financial applications.

DOI: https://doi.org/10.54660/.IJMRGE.2022.3.6.997-1000

Keywords: User Acceptance Testing (UAT), JSON Payload Validation, Postman Test Automation, Cloud-Based Billing Systems

1. Introduction

Could base billing systems are able to perform automated financial processing across decentralized models. These systems deal with flexible subscriptions, real-time tax adjustments, and compliance over bills in hybrid Azure environments. Upon the functional analysis, User Acceptance Testing (UAT) becomes the ultimate test to ensure that the business operations comply with the basic billing operations. In hybrid setups, UAT confirms billing accuracy by testing local dependencies, as well as cloud-based services, before production is deployed. Application Programme Interface testing improves accuracy of billing in terms of data transfer between services. Payloads are also checked by a structure integrity inspection to verify that transactions are correct and the taxes, discounts, and adjustments are reflected in the payload. Postman automation enhances the validation of the process at the UAT phases. The research will enhance the correctness of the UAT in billing apps situations by verifying the JSON payloads. To determine the effect of Postman automation on quality of billing verification in hybrid Azure environments. To develop validation processes of JSON formats that guarantee the reliability of billing, tax and compliance data. To determine the role of automated UAT in decreasing the manual testing and improving stakeholder confidence in the billing processes.

2. Literature Review

Particularly, in hybrid Azure environments, cloud billing systems are tailored to accommodate subscription model, the need to charge differently, tax laws, and compliance needs. APIs are also very significant in this architecture to enable complex interactions between financial modules with data.

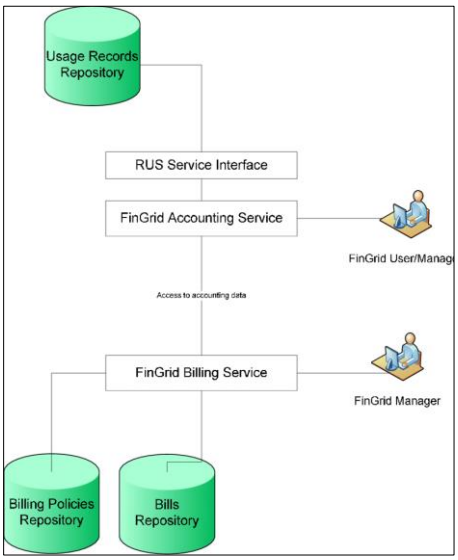

Fig 1: Cloud Billing Architecture

These environments are characterized by the variety of services, increasingly large amounts of data, and various regulations, and User Acceptance Testing (UAT) becomes a challenge in those conditions [1]. Research indicates that manual UAT is not enough to test the complicated billing principles, which may result in omitted finances and audit difficulties.

The characteristics, which make Postman so popular, include the possibility to test APIs, as it can request it to be sent automatically, check whether it was received, and interact with the CI/CD pipelines. Postman allows one to create collections, environment variables and JavaScript scripting in large organizations which make it easier to reuse tests [2]. It is reported that Postman in UAT identifies more faults since it emulates real users and audits billing endpoints.

The data sent by billing APIs are mostly in Nightjack format which includes things like transactions, taxes and updates. Validation is concerned with ensuring that the JSON is consistent with its schema, data types, and structure. The Postman scripts provide additional rules on what is anticipated using JSON schema, regular expressions and structure verifications [3].

Previous research on the testing of financial applications notes that automation is essential to achieve rule requirements and performance requirements. They discover that automated UAT, particularly checking APIs, is time-saving and is more audit ready [4]. Studies on banking and telecom billing have indicated that the look into JSON during automated testing increases the confidence that financial information is correct.

3. Research Methodology

The research presents an empirical study of IT case study to increase User Acceptance Testing (UAT) of cloud billing application developed within a hybrid architecture in Azure environments [5].

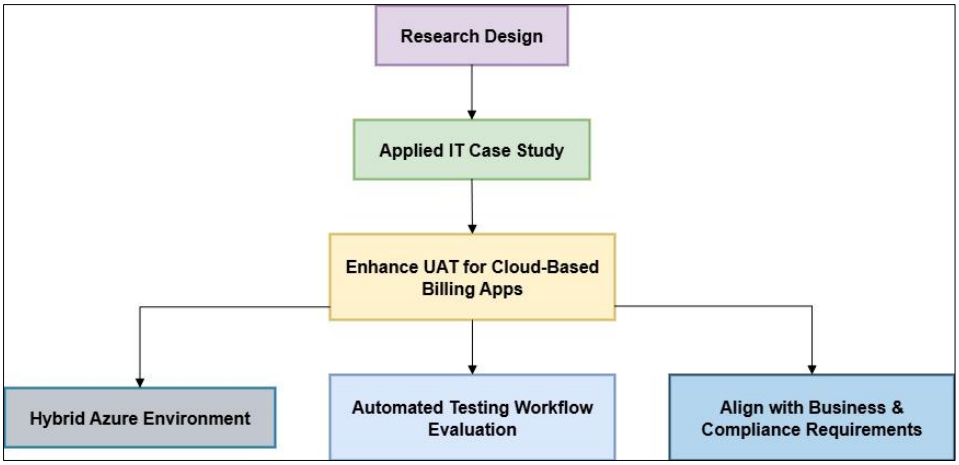

Fig 2: Research Design for Applied IT Case Study

The framework facilitates the assessment of the automated testing approach, besides fulfilling the business requirements, as well as the regulatory requirements. The main automation tool is Postman, which allows the organization of API testing by use of collection-driven executions. It displays the validations of the JSON schema to maintain the structure of the data associated with billing including tax details, billing modifications, and transaction data. Azure is the one that manages the deployment environment including the hybrid aspects of on-premises and cloud-based financial services.

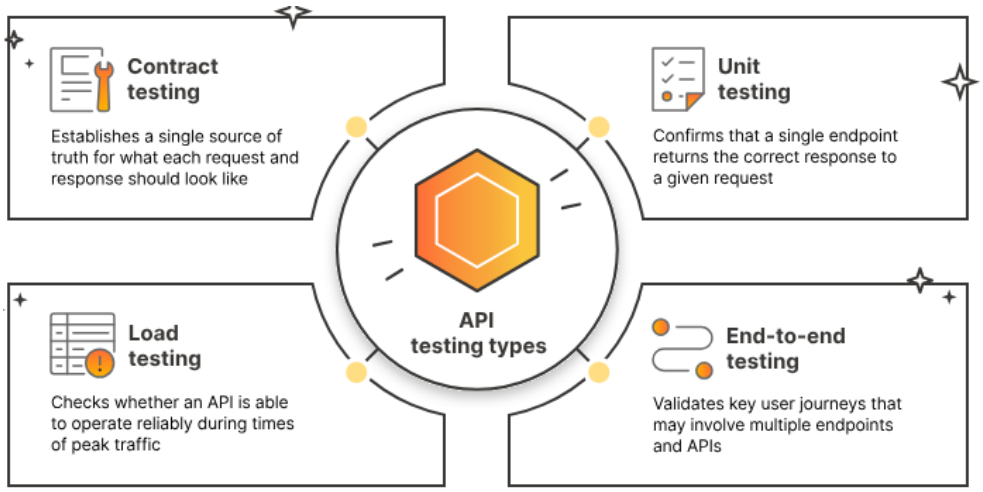

Fig 3: Postman in API Testing Frameworks

The UAT cases represent a wide range of business cases, which is possible to evaluate by real financial processes. The metrics of evaluation are accuracy of responses, the rate at which the schema is obeyed, the speed of error detection, and the decrease in manual verification. The focus of the scope is on billing precision, data quality and end-to-end automation accuracy [7].

5. Results and Analysis

The use of JSON payload in Postman to verify the payment of the billing was much more accurate during the UAT phase. The mismatch of tax calculations, currency exchange rates, and discount offers was detected earlier by implementing JSON schema checks on API responses. This decreased the chances of having anomalies that would go unnoticed into production [8]. Validation scripts were used to verify with regularity nested objects, where required billing attributes, including customer identifiers, invoice dates, and regulatory compliance tags were verified to exist and have the correct formats.

The automation strategy lessened the reliance on manual repetitive checks. Historically, QA engineers and business users cross-verified invoice entries and tax breakdowns with expected outputs manually. Postman automation replaced these tasks with parameterized test scripts capable of executing numerous billing scenarios within minutes [9]. This optimization reduced execution time by 60 to 70 seconds, allowing resources to focus on exploratory and compliance-related validations instead of substance-level verification.

The simultaneous checking of Postman and JSON payloads accelerated the UAT cycle without compromising accuracy. Organized validation was employed to eliminate uncertainty by applying consistent criteria for assessing test cases. This minimized false positives and established real-time feedback loops between developers, the QA team, and business stakeholders. The result was faster identification and resolution of bugs, which directly influenced shorter release cycles. The method faced difficulties, particularly in handling dynamic payloads generated by the cloud-based billing systems. The evolution of schemas across API versions necessitated frequent updates to the validation rules [10].

6. Discussion

The integration of JSON payload validation in Postman enabled alignment of the business requirement with the backend billing processes [11]. Stakeholders were able to validate that the system's outputs met the contractual and compliance needs by confirming the calculations of taxes, discounts, and adjustments through direct structured API responses.

The conventional UAT models are based on manual test execution, fixed spreadsheets and sample transaction review. These techniques do not tend to be deep in the validation of nested API structures and as such, anomalies are not identified [12]. Postman automation featuring JSON schema test achieved high accuracy, test coverage, and reduced human error, compared to traditional processes.

Table 1: Comparative Analysis – Traditional UAT vs. Automated UAT with Postman

Aspect	Traditional UAT	Automated UAT with Postman + JSON Validation
Execution Method	Manual test cases, spreadsheets, and invoice sampling	Automated test collections with reusable JSON schema checks
Accuracy of Billing Checks	Prone to human error; limited coverage	High precision with consistent structural validation
Test Coverage	Limited to sampled transactions	Wide coverage of multiple billing scenarios with parameterized tests
Time Efficiency	Time-intensive, requiring days for full cycle	60–70% faster execution with automated scripts
Stakeholder Confidence	Ambiguity due to subjective interpretation of results	Higher confidence through objective, repeatable verification

7. Conclusion and Future Work

The research showed that Postman-based automation combined with JSON payload validation contributed significantly to UAT accuracy of cloud-based billing systems to ensure proper validation of financial transactions, tax regimes, and other compliance logic. This solution minimized the use of the human element and harmonized the operations of the back office with business requirements. Future work may build on automation with AI-assisted anomaly detection on JSON responses and smooth CI/CD pipeline integration, allowing continuous testing of billing APIs and enhancing scalability with hybrid Azure environments.

References

1. Bhatta G, Pandey M. A case study on hybrid cloud approach to automate the cloud services based on decision support system. The Genesis. 2021 Sep;8(3):49-64. doi:10.47211/tg.2021.v08i03.010.
2. Tanwar J, *et al*. Project management for cloud compute and storage deployment: B2B model. Processes. 2022 Dec;11(1):7. doi:10.3390/pr11010007.
3. Ayyoub B, Zahran B, Nisirat MA, Al-Taweel FMS, Al Khawaldah M. A proposed cloud-based billers hub using secured e-payments system. TELKOMNIKA (Telecommunication Computing Electronics and Control). 2021 Feb;19(1):339. doi:10.12928/telkomnika.v19i1.15879.
4. Demircioğlu ED, Kalipsiz O. API message-driven regression testing framework. Electronics. 2022 Aug;11(17):2671. doi:10.3390/electronics11172671.
5. Mishra DP, Rout KK, Salkuti SR. Modern tools and current trends in web-development. Indonesian Journal of Electrical Engineering and Computer Science. 2021 Nov;24(2):978-985. doi:10.11591/ijeecs.v24.i2.pp978-985.
6. Lv T, Yan P, He W. Survey on JSON data modelling. Journal of Physics: Conference Series. 2018 Aug;1069:012101. doi:10.1088/1742-6596/1069/1/012101.
7. Şimşek U, Kärle E, Fensel D. Machine readable web APIs with Schema.org action annotations. Procedia Computer Science. 2018;137:255-261. doi:10.1016/j.procs.2018.09.025.
8. Walonoski J, Scanlon R, Dowling C, Hyland M, Ettema R, Posnack S. Validation and testing of fast healthcare interoperability resources standards compliance: data analysis. JMIR Medical Informatics. 2018;6(4):e10870. doi:10.2196/10870.
9. Mansouri Y, Prokhorenko V, Babar MA. An automated implementation of hybrid cloud for performance evaluation of distributed databases. Journal of Network and Computer Applications. 2020 Oct;167:102740. doi:10.1016/j.jnca.2020.102740.
10. Rodriguez MA, Buyya R. Container-based cluster orchestration systems: a taxonomy and future directions. Software: Practice and Experience. 2019;49(5):698-719. doi:10.1002/spe.2660.
11. Alanda A, Mooduto HA, Hadelina R. Continuous integration and continuous deployment (CI/CD) for web applications on cloud infrastructures. JITCE (Journal of Information Technology and Computer Engineering). 2022 Sep;6(2):50-55. doi:10.25077/jitce.6.02.50-55.2022.
12. Donca IC, Stan OP, Misaros M, Gota D, Miclea L. Method for continuous integration and deployment using a pipeline generator for agile software projects. Sensors. 2022 Jun;22(12):4637. doi:10.3390/s22124637.