WHITEPAPER

Transforming Quality Assurance Approach for Product Excellence

The Evolving Practices and Technologies in Modern Software Testing

Gleecus TECHLABS INC

Forward Thinking Digital Innovation

Table Of Contents

Executive Summary

This whitepaper delves into the profound impact that evolving technologies, such as artificial intelligence, machine learning, generative AI, and cloud computing, introduce into the Quality Assurance (QA) processes within a typical software development lifecycle.

> *The integration of AI and machine learning in QA processes is not just enhancing efficiency but also transforming how we approach software testing altogether. - Gartner*

Within the scope of this paper, we embark on an exploration of various facets, including how the role of the test plan extends beyond the realm of the user interface. Additionally, we delve into strategies for mitigating technical debt through the adoption of a "shift left" approach, which involves the early detection and rectification of flaws and vulnerabilities at the inception of the development phase.

In a software development landscape that is progressively becoming more inclusive and accessible, thanks to the rise of low-code approaches, the demand for a transformation in the way test scripts are authored becomes evident. We investigate this transition from the era of labor-intensive test script coding to the creation of more user-friendly Business Validation Tests (BVT). This shift empowers non-technical stakeholders, including product managers and C-suite executives, to actively engage in the testing process.

Throughout this whitepaper, we provide valuable insights into how organizations can strategically adopt cutting-edge technologies to enhance their QA practices. These enhancements serve as the robust foundation for delivering superior IT products and services.

Quality Assurance as the Bedrock of IT Products and Services

In today's rapidly evolving digital landscape, the critical role of QA cannot be overstated. QA serves as the linchpin, the underpinning, and the bedrock upon which the success and reliability of IT products and services are built. To truly grasp the magnitude of its significance, one must look beyond the surface and into the realm of facts and statistics that paint a compelling picture of QA's pivotal role in the modern world of technology.

Recent statistics underscore the increasing reliance on digital solutions. As of the last available data, there were a staggering 4.9 billion active internet users globally, a number representing over 60% of the world's population. Mobile device usage continues its meteoric rise, with more than 5.2 billion unique mobile users. In the wake of the COVID-19 pandemic, remote work and online commerce have become ubiquitous, with a 44% increase in e-commerce sales worldwide in 2020 alone.

These trends are indicative of a digital transformation that has accelerated at an unprecedented pace. In this context, the quality and reliability of IT products and services are no longer mere aspirations but essential requirements for businesses, governments, and individuals alike.

In the ever-evolving landscape of software development, where innovative technologies such as Cloud, Analytics, AI, ML Automation and third-party tools are seamlessly integrated, the bar for quality standards has been set high. Users now demand nothing less than flawless performance and functionality, whether the software serves external customers or internal operational needs. In response, businesses are tasked with the critical mission of ensuring the impeccable functionality of every facet of their software, delivering nothing short of exceptional customer experiences.

Beyond the realm of customer satisfaction, a robust Quality Assurance (QA) framework presents substantial advantages for the software product's operational, developmental, and business development facets, positioning it as an indispensable asset in delivering unparalleled product value.

Foremost among the virtues of QA lies its capacity to unearth bugs and errors during the early phases of development, epitomizing the "shift-left" approach. Through comprehensive and meticulous testing, QA experts proficiently pinpoint issues encompassing reliability, functionality, usability, portability, security, and compliance. This proactive QA methodology paves the way for the expedient resolution of these issues, thereby averting the accrual of technical debt.

Moreover, a diligently executed QA regimen engenders the creation of software that is both steadfast and competitive. By perpetually subjecting the software to rigorous tests and validations across diverse technical landscapes, QA professionals substantially enhance its overall quality, ensuring not only that it aligns with but surpasses customer expectations.

Impact of Generative AI and LLM on Various Industries

Testing in today's environments is more complex than a few years ago. The more complex the frontend features get the more intricate the backend architecture becomes. Application complexity gives rise to handling of high frequency transactions that can be aptly supported by advanced databases like cloud. Protecting data integrity mostly through a cloud infrastructure to synchronize the collection and retrieval of data is another important aspect. To gain speed and improve fault tolerance applications today come as a pack of distributed services in a microservice based architecture creating an increased complexity, overhead, and friction in testing. The specialized need of testing gives rise to separate areas of testing and warrants a lot of preparation, infrastructure building, tech integration, and maintenance.

Database Testing

Database testing plays a crucial role in ensuring the integrity and quality of data stored within a database system. It involves various testing methodologies and techniques to validate the structural, functional, and non-functional aspects of the database.

ACID properties

One of the fundamental aspects of database testing is evaluating the compliance with ACID properties (Atomicity, Consistency, Isolation, Durability). These properties ensure that transactions are processed reliably and consistently.

Techniques to check ACID properties involve comprehensive testing strategies. Unit tests evaluate individual components to verify atomicity. Functional and integration tests validate consistency by ensuring data remains valid through various operations. Isolation is ensured through concurrency and multi-user testing, simulating real-world scenarios. Durability is confirmed by subjecting the system to failure and recovery tests, ensuring committed transactions survive system crashes. Through these techniques, databases can be thoroughly evaluated to adhere to the essential ACID properties, promoting robust, reliable, and consistent transactional behavior.

Server validation

Database server validation is performed to ensure that the server configuration aligns with the desired specifications. This includes validating security measures, performance optimization settings, and backup mechanisms.

White Box and Black Box Testing

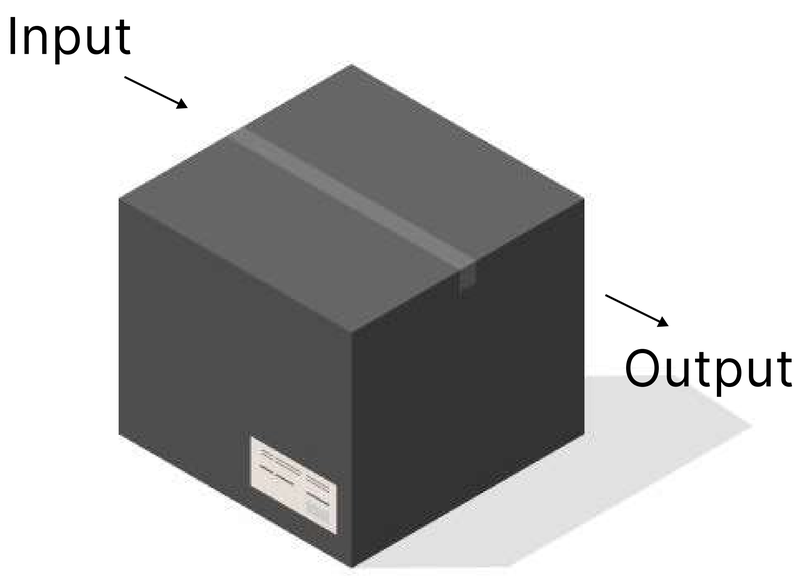
White box and black box testing are functional testing approaches to check if all the expected functionalities of a database system are met.

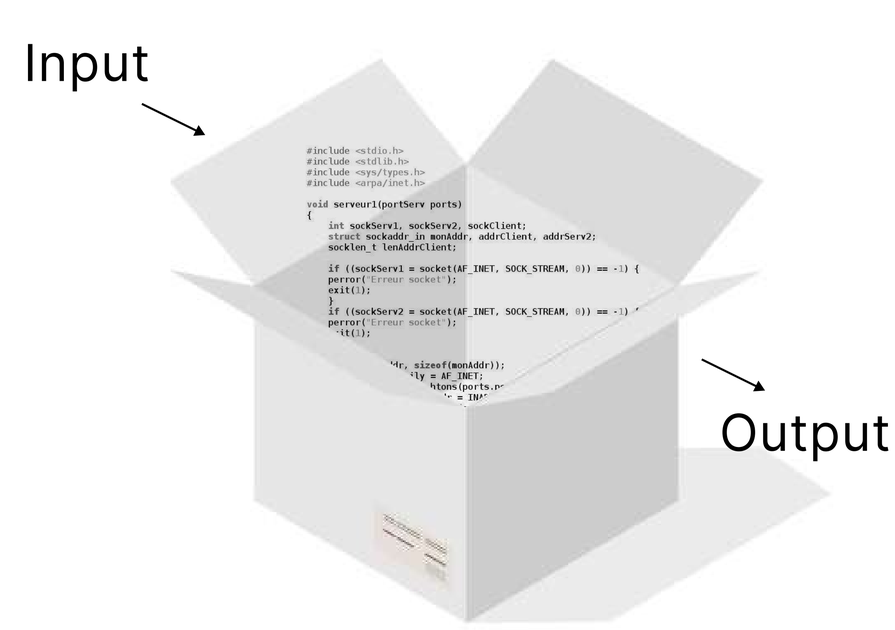
Black Box Approach

The white box approach, also known as structural or glass box testing, involves examining the internal logic and structure of the database. Testers have detailed knowledge of the database architecture, schema, and code. They design tests based on this understanding to validate the accuracy of SQL queries, stored procedures, triggers, and other components. This approach aims to uncover vulnerabilities, such as improper data handling and security breaches, by analyzing the database's inner workings.

White Box Approach

The white box approach, also known as structural or glass box testing, involves examining the internal logic and structure of the database. Testers have detailed knowledge of the database architecture, schema, and code. They design tests based on this understanding to validate the accuracy of SQL queries, stored procedures, triggers, and other components. This approach aims to uncover vulnerabilities, such as improper data handling and security breaches, by analyzing the database's inner workings.





Hidden or unknown	**Internal structure**	Known
Not needed	**knowledge of implementation**	Needed
Not required	**knowledge of programming**	Required
Functional behavioral test	**Type of test**	Structural, logic test
On the basis of req. spec document	**Testing initiated**	After detail design document
Higher levels	**Level of software testing**	Lower levels

Apart from this, non-functional tests like load and stress testing simulate real-world scenarios to evaluate how well the database handles high volumes of concurrent users or excessive workloads.

Ultimately, thorough database testing helps identify potential issues early on to maintain data integrity, optimize performance, and enhance user experience. By adhering to industry best practices in this area, organizations can confidently rely on their databases for critical operations while ensuring high-quality data management.

Microservices Testing

Microservices testing is a critical aspect of ensuring the reliability and functionality of a microservices architecture. It involves various types of tests tailored to the unique characteristics of microservices, each serving a specific purpose in validating the system's behavior and performance.

Unit tests

Microservices are designed to be independent and isolated, making unit tests crucial to validate each microservice's functionality at a granular level in isolation. Unit tests focus on validating individual units of code, such as functions, methods, or small components, in isolation.

Contract tests

Contract tests validate the contracts or agreements between different microservices. These tests ensure that the interactions and integrations between microservices align with the specified contracts, preventing any regressions when changes are made.

Integration tests

Integration tests verify the interactions and integrations between multiple microservices, ensuring that they function seamlessly when combined. These tests are essential for identifying issues that may arise when integrating various microservices together.

End-to-End tests

End-to-end tests evaluate the entire microservices ecosystem, simulating real-world user scenarios. These tests ensure that the microservices work together harmoniously to deliver the intended functionality and user experience across the entire system.

Cross browser testing, a vital component of end-to-end testing, ensures that the microservices-based application functions consistently across different web browsers. Cross browser testing tools allow testing on a wide range of browsers and operating systems to guarantee a consistent user experience regardless of the user's browser choice.

Cloud Services Testing

Cloud testing has become an integral part of software development and deployment. With the rise of cloud computing, organizations are leveraging various "as-a-Service" (aaS) offerings like Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS) to streamline their testing processes.

One of the key reasons for cloud testing is to check integration of an application to a cloud environment, specifically for SaaS applications. This also benefits testers to simulate real-world scenarios and assess the performance and scalability of their applications under different user loads.

Multi-tenancy testing

Multi-tenancy testing is another important aspect of cloud testing. As SaaS applications cater to multiple users simultaneously, it is crucial to ensure that data isolation and security measures are in place. By conducting thorough multi-tenancy testing, organizations can validate the effectiveness of their application's architecture in maintaining data separation between different tenants.

Disaster recovery testing

Disaster recovery testing is a critical component when it comes to cloud-based applications. With the ability to quickly replicate environments and data backups, organizations can test their disaster recovery plans effectively within a cloud environment. This ensures that business-critical systems can be restored promptly in case of any unforeseen events.

Furthermore, as more organizations adopt cloud-native software development methodologies, it becomes crucial to utilize specialized tools designed for cloud-based environments. Cloud-based tools offer features such as network testing and security testing specifically tailored for the unique challenges posed by distributed systems and virtualized infrastructure.

API testing

API testing plays a crucial role in ensuring the functionality and reliability of software applications. It involves testing the various aspects of an application programming interface (API), including functional testing, performance testing, and reliability testing.

Functional testing

Functional testing focuses on verifying that the API behaves as expected in different scenarios. This includes testing edge cases and boundary values to ensure that the API handles all possible inputs correctly. By thoroughly examining the API's responses to different inputs, testers can identify any issues or inconsistencies in its behavior.

Performance testing

Performance testing is another important aspect of API testing. It involves evaluating the response time and throughput of an API under different loads and stress conditions. By measuring these metrics, testers can assess how well the API performs under heavy usage and identify any potential bottlenecks or performance issues.

Reliability testing

Reliability testing is also a critical part of API testing. Testers simulate error scenarios such as network failures or server downtime to ensure that the API handles such situations gracefully. This includes verifying that appropriate error codes are returned, error handling mechanisms are in place, and proper documentation is provided to guide developers on how to handle errors.

Documentation review

Thorough documentation review is essential for successful API testing. Testers need to review the documentation provided for the API and validate the accuracy and completeness of the API documentation, including endpoint descriptions, request/response examples, authentication methods, and error handling details.

IoT Testing

IoT testing ensures readiness for real-world applications by assessing an IoT solution through various tests. Its goal is to identify and address vulnerabilities, instilling confidence in its expected functionality upon deployment.

Key IoT testing features comprise:

Device testing

Involves connecting a device to a mobile or computer, running software, and performing necessary checks—simulating a virtual machine on the device. This process consists of a series of testing that helps in validating the functionality, performance and security of the devices and the software in the IoT system.

Emulator testing

Utilizes emulators running on different machines, providing a test environment for applications. This approach proves valuable when developing apps for diverse devices, allowing testing without setting up unique development environments for each device.

Understanding hardware compatibility for specific tests is vital in emulator testing. IoT testing is an evolving domain necessitating diverse testing strategies.

Technologies and Processes Defining the Modern QA Approach

Chaos Engineering

Chaos engineering involves deliberate disruption of a system through experimental processes to pinpoint vulnerabilities, foresee failures, anticipate user experience, and improve system architecture. This practice assists engineering teams in improving the resilience of an organization's infrastructure by reconfiguring and reinforcing it. The driving forces behind the adoption of chaos engineering are mounting consumer expectations and growing system complexity. During the chaos testing the application being tested is subjected to known failures within a specific radius and any abnormalities are noted and developers work on checking the vulnerabilities. Typically a web application is vulnerable to four types of attacks. Chaos engineering simulates these attacks in a controlled manner.

Resource attack

Resource attack involves starving the system of resources like CPU, memory, I/O, or disk space. It measures how the service degrades when a system is denied resources.

State attack

This involves disrupting state within the architecture and check if the service fails or is able to handle it.

Network attack

This test involves tampering with the network causing data loss or delay in traffic. This test is done to infer how the system reacts when they are deprived of accessing certain dependencies like third-party services or APIs.

Application attack

This kind of testing involves overloading the traffic to an application and putting the application under strain. The objective of this kind of load test is to validate the scalability of an application.

CASE STUDY

NETFLIX

Netflix pioneered the use of Chaos Engineering to ensure the resilience of their microservices architecture. By deliberately introducing failures into their production environment using their tool, Chaos Monkey, they identified weaknesses and improved their system's robustness.

Behavior Driven Testing (BDT)

BDT or behavior driven testing is a software development and testing methodology where the tests are written from a user perspective and behavior. Tests are written using domain-specific language (DSL) in a human readable syntax to emphasize collaboration and communication among various stakeholders, including developers, QA teams, product managers, and non-technical stakeholders.

Principles of BDT

BDT is based on the following principles:

- **Collaboration**: BDT encourages collaboration between developers, testers, and business stakeholders. This helps to ensure that everyone is on the same page and that the tests are aligned with business needs.

- **Communication**: BDT uses a domain-specific language to write tests in a way that is easy to understand for everyone involved in the development process. This helps to improve communication and reduce misunderstandings.

- **Behavior**: BDT focuses on testing the behavior of the software, rather than the implementation. This makes the tests more robust and easier to maintain.

Given-When-Then (GWT) syntax

- **Given**: Describes the preconditions or the initial context before the behavior is exhibited.
- **When**: Specifies the actions or events that trigger the behavior.
- **Then**: Defines the expected outcomes or results after the actions in the "When" step are performed.

Cucumber for BDT

Cucumber's salient features for Behavior-Driven Testing (BDT) include:

- **Human-Readable**: Gherkin syntax allows non-technical stakeholders to define behavior.

- **Collaborative**: Facilitates teamwork and shared understanding of software behavior.

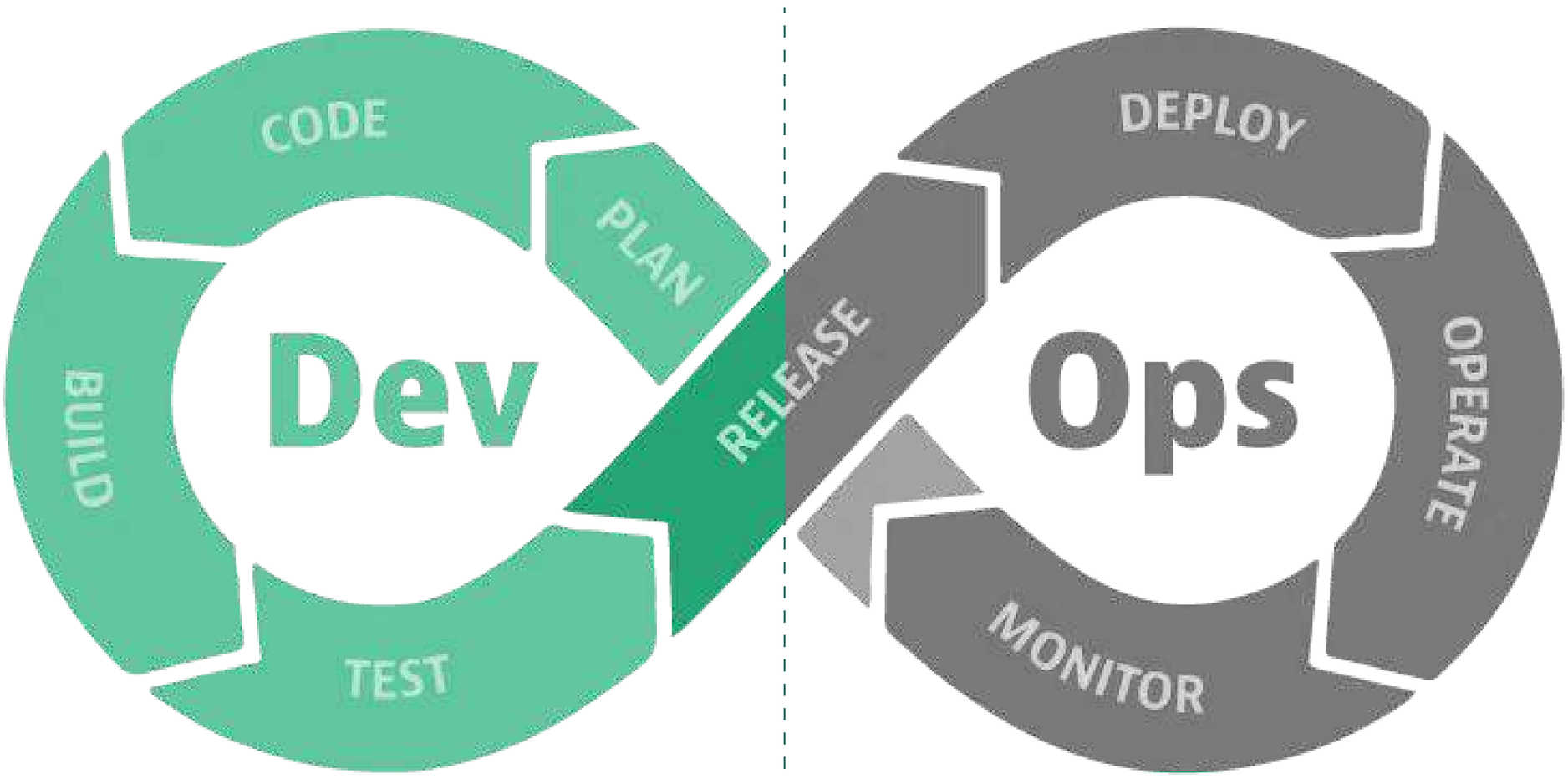
- **Executable Documentation**: Specifications are converted into automated, living documentation.

- **Integration-Friendly**: Easily integrates with diverse technologies and frameworks.

- **Traceability**: Links Gherkin steps to executable code, ensuring comprehensive coverage and clarity.

Other popular BDT frameworks are SpecFlow, Quantum, JBehave etc.

Shift-Left Approach



Shift Left

Ensures software meets design

Shift Right

Ensures performance, resilience, reliability

Shift-left testing is when the testing is conducted on the phases to the left side of the symbolic representation of the software development cycle as an infinite loop. This means testing, quality, and performance evaluation are performed right at the beginning phases of planning, design, and often even before code is written. Shift left thus helps to understand whether the software even meets customer requirements at the early stages of development. Accordingly, changes can be implemented way ahead before the software moves into production.
Shift left can be categorized into four types

Traditional shift left testing

Focuses on early unit and integration testing, emphasizing APIs and cross-browser compatibility. Emphasizes delivering information in smaller, early unit-tested code portions rather than broader system-level operational and acceptance testing.

Incremental shift left testing

Gains popularity for teams transitioning from waterfall to smaller, incremental project development. Involves systems-level operational and acceptance testing in smaller, incremental phases, making it effective for validating large and complex systems.

Agile/DevOps testing

Involves continuous, short sprints during development without addressing operational performance. Validates adherence to basic requirements architecture in an agile or DevOps development environment.

Model-Based shift left

Aims to reduce errors introduced in requirements definition, architecture, and design phases. Tests for executable requirements, architecture, and design, allowing testing to begin almost immediately rather than waiting for completion of all other test cycles.

CASE STUDY

Google

Google adopted a shift-left testing approach to enhance the quality and speed of their software development process. By integrating testing early in the development cycle, they significantly reduced bugs and improved time-to-market.

Generative AI

Generative AI has the capability to build comprehensive plans, generate large volumes of data, and explore data beyond human speed. It finds distinct implementations at every stage of the software development lifecycle.

Requirement phase

Generative AI is a powerful tool for requirement analysis and test coverage in software testing. Through automated requirement generation, it can analyze existing artifacts and produce new or refined requirements, ensuring a comprehensive set. By understanding the semantic meaning of requirements, AI can enhance clarity and consistency while identifying conflicts for resolution. Leveraging historical project data, it identifies patterns and suggests potential testing areas, aiding in more targeted testing efforts.

Test case development

Generative AI significantly aids in test case development by automating the generation of diverse, complex test scenarios. By leveraging machine learning algorithms, it analyzes existing test cases, identifies patterns, and creates new ones. This process enhances test coverage, detecting potential issues in software. Generative AI also helps in load testing, simulating thousands of users and identifying system limitations. Furthermore, it accelerates agile development by rapidly generating test cases for new features, improving efficiency and allowing QA teams to focus more on strategic testing and ensuring high software quality. Ultimately, generative AI revolutionizes test case creation, optimizing the testing process and enhancing overall product reliability.

Generative AI for BDT

Since BDT uses human understandable language to write test cases, generative AI finds a significant contribution to writing realistic, diverse, and compliant test data. Gathering original test data might be difficult and time taking, generative AI performs data synthesis to create synthetic data from scratch, based on predefined rules, models, or patterns. With proper prompts even from a non-technical person, generative AI can create test scripts in a very short time.

CASE STUDY

Salesforce utilized generative AI to automate test case development, resulting in improved test coverage and faster development cycles. This approach enabled non-technical stakeholders to contribute to the testing process, enhancing overall product quality.

DevSecOps

DevSecOps, an extension of the DevOps approach, integrates security practices into the software development and delivery process, aiming to ensure security is a shared responsibility and is built into the software from the onset of SDLC. When it comes to testing, DevSecOps offers several benefits.

Continuous monitoring

DevSecOps promotes continuous monitoring of applications and infrastructure in real-time. Security testing tools continuously monitor the application for potential vulnerabilities, enabling rapid response to emerging security threats.

Threat modeling and risk assessment

DevSecOps encourages threat modeling and risk assessment during the planning and design stages. Security considerations are integrated into the development process by identifying potential threats and assessing their impact on the application.

Security as code

Security controls and requirements are treated as code, enabling versioning, peer reviews, and automation. Security configurations are managed alongside application code, ensuring consistent and secure deployment.

ML In Testing

Machine Learning (ML) offers several ways to enhance software testing processes, making them more efficient, effective, and accurate.

Test case generation

ML algorithms can analyze application code and automatically generate test cases, identifying potential scenarios that may not be apparent to human testers. This accelerates the test case creation process and improves test coverage.

Predictive analysis

ML models can predict areas of the application that are more prone to defects based on historical test data, helping testers focus their efforts on critical areas and prioritize testing accordingly.

Bugs and anomalies detection

ML can detect anomalies or unexpected behaviors in the application, helping identify potential bugs or areas that require further investigation. This anomaly detection contributes to robust test coverage.

Test prioritization

ML can prioritize test cases based on various factors like code changes, historical defect data, and business impact, ensuring that critical areas of the application are thoroughly tested first.

Log analysis

ML algorithms can analyze logs and identify patterns related to errors or issues, assisting in diagnosing problems and troubleshooting more efficiently.

Test Automation

Test automation has revolutionized the software testing landscape, offering efficiency, repeatability, and speed in testing processes. Two prominent approaches within test automation are Robotic Process Automation (RPA) testing and Scriptless Automation Testing.

RPA testing

Robotic Process Automation (RPA) mimics human interactions with software systems, automating repetitive tasks and processes. In the context of testing, RPA bots can replicate user actions and interactions with an application, allowing for end-to-end testing of workflows. RPA testing helps ensure software applications perform as expected from a user's perspective by automating actions like data input, form submissions, and navigation. It enables faster testing cycles and frees testers to focus on more complex scenarios.

Scriptless automation testing

Scriptless automation, also known as codeless or no-code automation, enables testers with limited or no programming skills to create and execute automated tests. This approach typically involves using intuitive graphical interfaces or pre-defined components to design test cases. Testers can create automation scripts by dragging and dropping elements, defining workflows, and configuring test steps. Scriptless automation accelerates the test creation process, reduces dependency on coding expertise, and promotes collaboration between technical and non-technical team members.

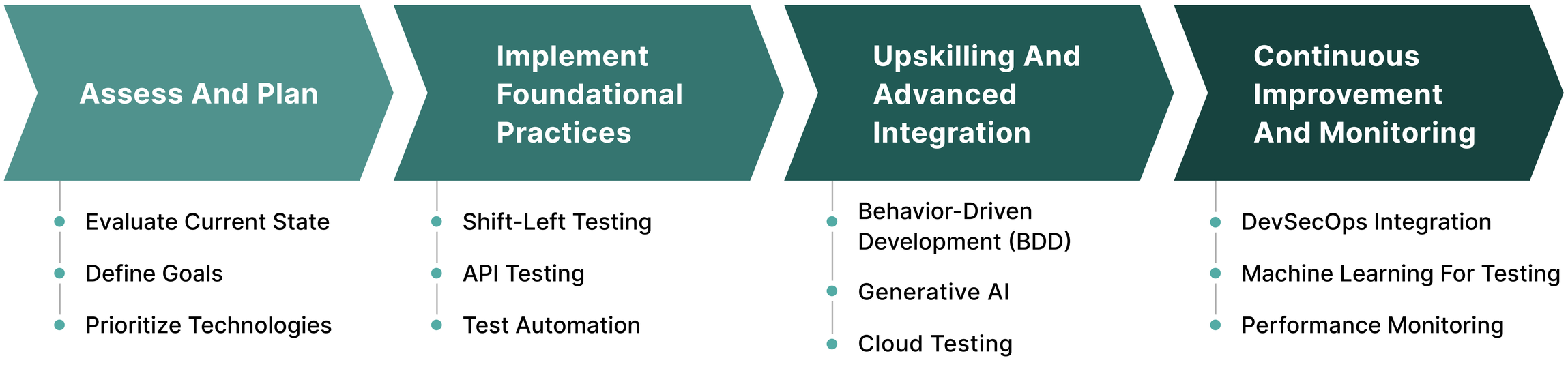
Conclusion

The evolving technologies discussed in this whitepaper offer a number of benefits to the QA process. By integrating these technologies to the QA practices, product engineering teams can improve the efficiency, effectiveness, coverage, and risk reduction of their testing efforts, leading to delivery of robust software in a very limited time.

Adoption to modernization in the QA process need not be attempted all at once. Businesses can start by adopting a few technologies that are most relevant to their needs and then gradually expand their adoption over time.

It is important to get buy-in from stakeholders, such as developers and business managers, before adopting new technologies. This will help to ensure that the new technologies are properly supported and that they are used in a way that benefits the entire team. Measuring the success of QA modernization effort can help to understand which technologies bring a major overhaul to the QA practices and encourage to explore more QA related cutting edge technologies.

A Roadmap to Adopting Modern QA Technologies and Practices



About Us

Gleecus TechLabs Inc. is an ISO 9001:2015 – Quality Management and ISO/IEC 20000-1:2018 – IT Service management certified IT innovation partner for startups, SMBs, and enterprises helping clients envision, build, and run more innovative and efficient businesses. We are an experienced member of the AWS partner network (APN) catering to a diverse range of Cloud, Data Engineering, AI, and Managed Services needs for our clients.

> **Our team empowers businesses to deliver high quality software for their users by modernizing the QA practices. Our QA roadmap continuously evolves, responding to your business needs and customer feedback by integrating the best-in-class solutions.**

We assess your existing state of QA and identify bottlenecks to set up a roadmap for phased adoption of advanced testing practices and technologies. We keep a continuous monitoring of the efforts and outcomes of the new practices and train your stakeholders to champion them.

Transform your testing practices adopting cutting-edge technologies for high quality software delivery

Connect with Us

About Gleecus TechLabs Inc.

Gleecus TechLabs Inc. is an ISO 9001:2015 and ISO/IEC 20000-1:2018 certified Forward Thinking Digital Innovation partner creating impactful business outcomes with Engineering & Experience. With deep focus on Cloud, Data, Product Engineering, AI and Talent we help organizations become Digital Natives.

Gleecus TECHLABS
Forward Thinking Digital Innovation

Email: hello@gleecus.com

Phone: +1 347 947 2022

www.gleecus.com