Trends in Software Testing

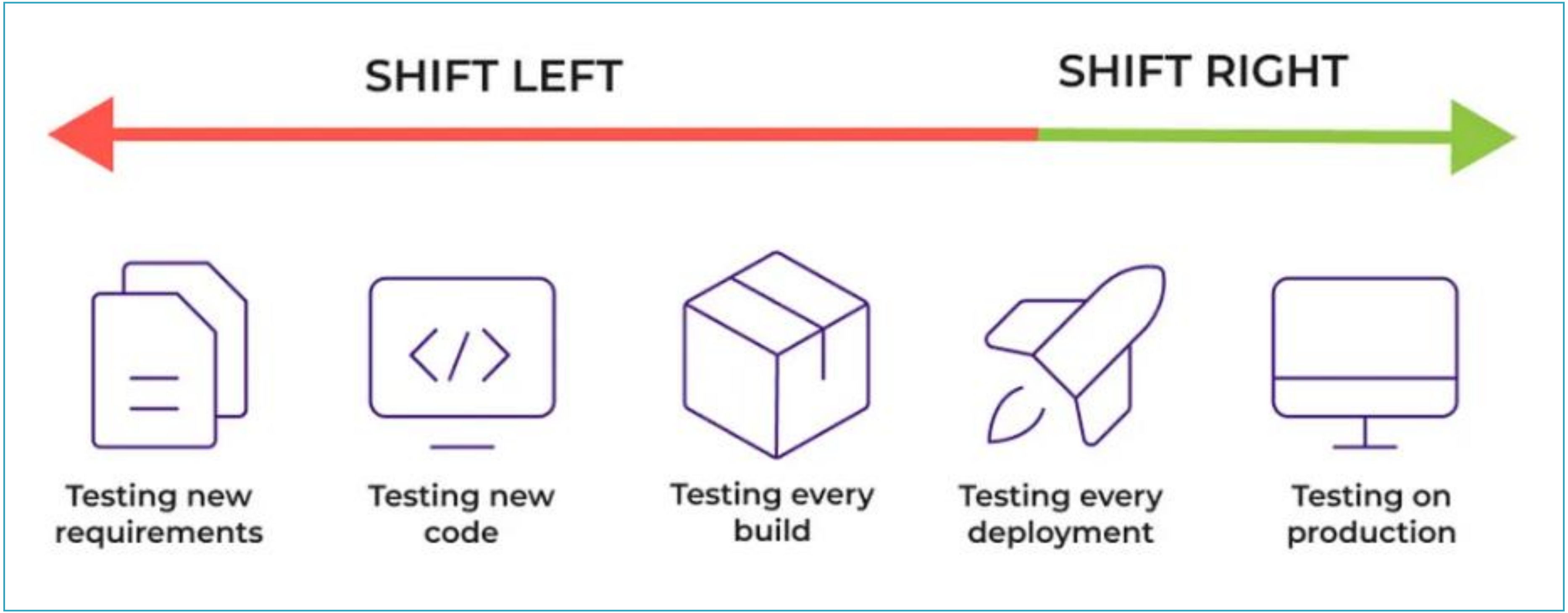
4th May 2025

Mr. Suresh Fernando

Trends in Software Testing

- Shift-Left Testing
- AI and ML in Testing
- Growing Use of QAOps
- Crowdtesting
- Enhanced Focus on Security Testing
- IoT Testing
- Mobile Test Automation
- Enhanced API Testing and Automation
- Focus on accessibility testing

1. Shift-Left Testing



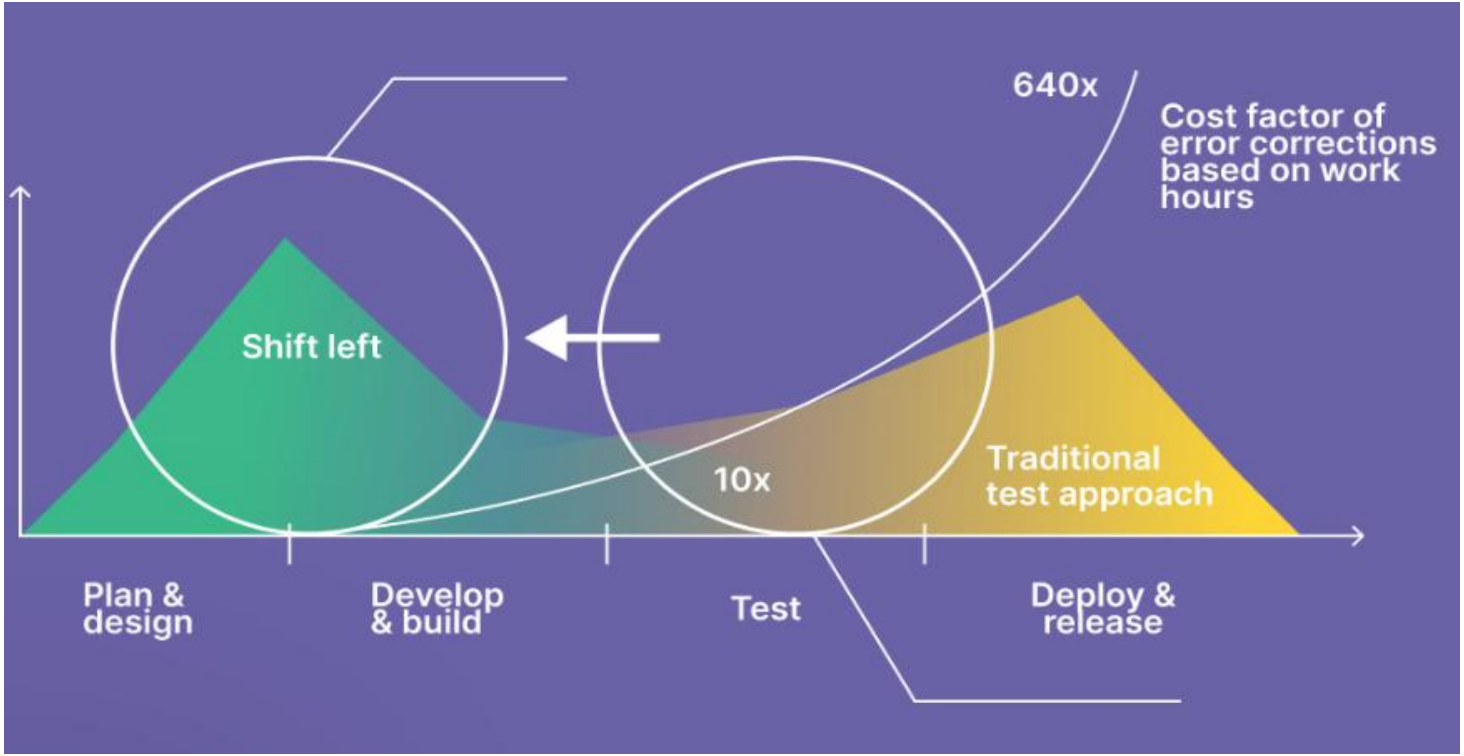
Shift-Left Testing

- Emphasizing early and frequent integration of testing in the software development cycle
- Why important?
  - Identifying and addressing issues sooner
  - Accelerating market time
  - Enhancing software release quality
  - Reducing the time spent on debugging
  - Enabling teams to devote more effort to feature and functionality enhancement

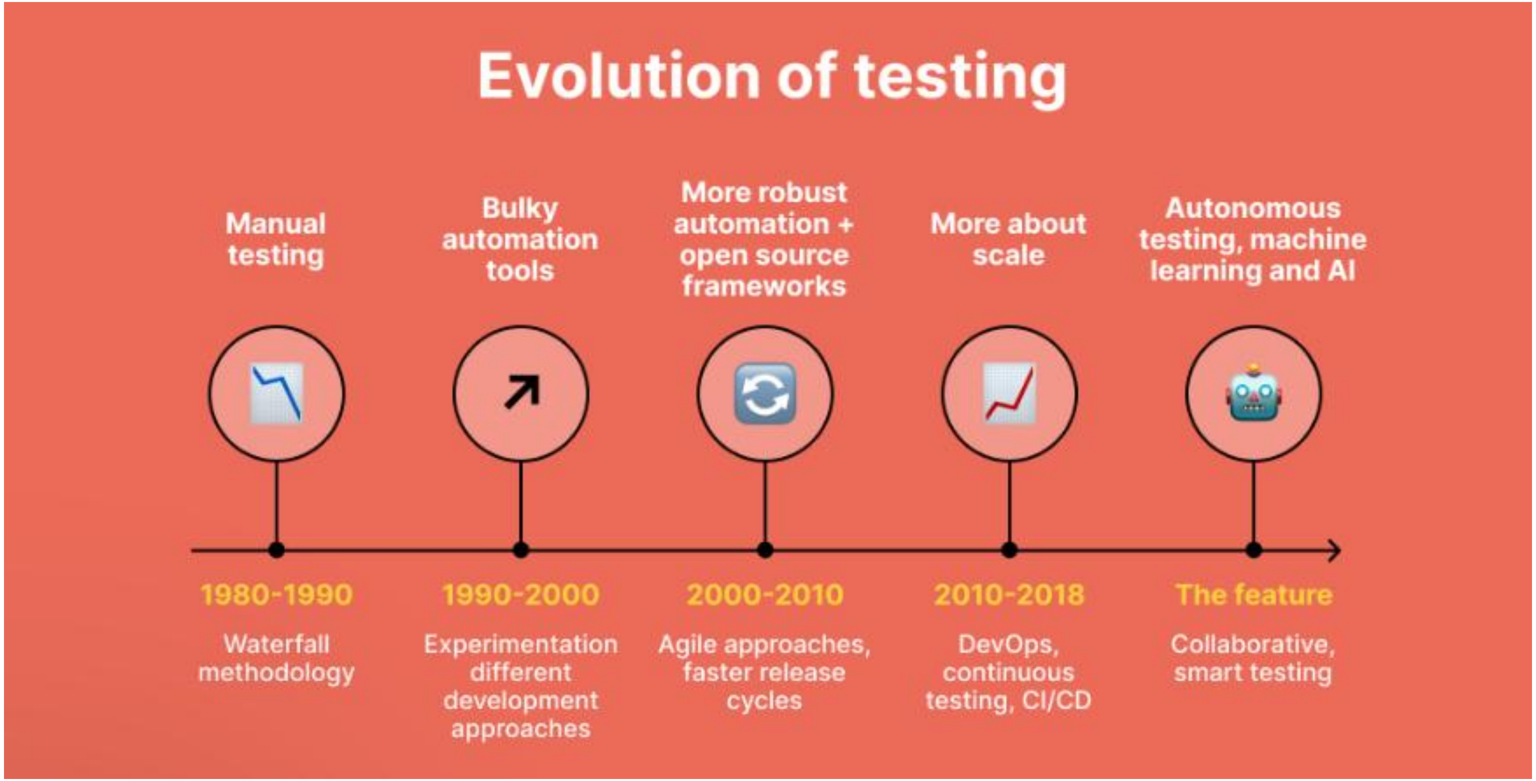
Shift-Left Testing

- Delayed testing will result in:
  - Insufficient testing resources
  - Missed design
  - Architectural or requirements flaws
  - Complexities in debugging and issue resolution, and
  - Project delays.

Shift-Left Testing


640x
Cost factor of error corrections based on work hours
Shift left
10x
Traditional test approach
Plan & design
Develop & build
Test
Deploy & release

2. AI and ML in Software Testing



Evolution of testing

Manual testing

1980-1990

Waterfall methodology

Bulky automation tools

1990-2000

Experimentation different development approaches

More robust automation + open source frameworks

2000-2010

Agile approaches, faster release cycles

More about scale

2010-2018

DevOps, continuous testing, CI/CD

Autonomous testing, machine learning and AI

The feature

Collaborative, smart testing

AI and ML in Software Testing

Test Case Generation: AI and ML algorithms can automatically generate test cases based on software requirements and historical data

Test Prioritization: AI can analyze code complexity, recent changes, and historical defect data to identify and prioritize the most critical test cases for execution.

Defect Prediction: By analyzing patterns in historical data, AI can predict potential areas of failure, allowing testers to focus their efforts on those areas.

Test Execution and Result Analysis: AI can automate the execution of test cases and analyze the results, identifying potential errors and defects.

Intelligent Feedback: AI can provide more accurate and context-aware feedback to testers, helping them understand the root cause of defects.

Benefits of Using AI and ML in Testing

Increased Efficiency: Automating tasks and prioritizing test cases saves time and resources.

Improved Accuracy: AI can help identify and fix defects more effectively, leading to higher quality software.

Cost Reduction: By automating tasks and reducing manual effort, AI and ML can lower the overall cost of software testing.

Enhanced Software Quality: By identifying and fixing defects early, AI and ML can help ensure that software meets quality standards.

Improved Developer Productivity: AI can help developers by providing them with feedback on their code and identifying potential issues before they become bugs.

Challenges of Using AI and ML in Testing

Data Requirements: AI and ML algorithms require large amounts of data to train and learn, which can be a challenge for some projects.

Algorithm Complexity: Developing and maintaining AI and ML algorithms can be complex and require specialized expertise.

Integration with Existing Tools: Integrating AI and ML tools into existing software testing workflows can be challenging.

Ethical Considerations: As AI and ML become more prevalent in software testing, it's important to consider the ethical implications of using these technologies, such as bias and fairness.

AI/ML Techniques in Software Testing

Natural Language Processing (NLP)
Understanding user stories, requirements, or test cases written in natural language
(Automatic test case generation | Requirement traceability)

Predictive Analytics
Using historical test data to forecast future defects or risky components
(Defect prediction | Test prioritization)

Clustering & Classification
Grouping test cases or defects by similarity or categorizing them
(Test suite optimization | bug triage)

Reinforcement Learning
Systems learn the best sequence of actions (e.g., in test generation) through trial and error
(Intelligent exploratory testing)

Anomaly Detection
Identifying patterns that deviate from the norm
(Monitoring in production – AIOps | detecting flaky tests)

NLP in Testing

- NLP in software testing is used to help automate, understand, or improve testing tasks that involve human language.
- **How NLP is used:**
  - **Analyzing requirements:** NLP can read written requirements (in English or other languages) and automatically suggest or generate test cases.
  - **Generating test cases from user stories:** By parsing user stories or acceptance criteria written in natural language, NLP tools can help draft test cases, reducing manual effort.
  - **Analyzing bug reports:** NLP can cluster, categorize, or prioritize bug reports by reading the language used - e.g. identifying duplicate issues or estimating severity.
  - **Automated testing of conversational systems:** For chatbots or voice assistants, NLP helps test whether the system understands and responds correctly to varied natural language inputs.

NLP in Testing - *An Example*

- Imagine a software team working on an e-commerce platform. The product team writes a user story:

  "As a customer, I want to add products to my shopping cart so I can purchase them later."

- An NLP-based test tool can read this sentence and suggest:
  - Test case 1: Verify adding a single product to the cart.
  - Test case 2: Verify adding multiple products.
  - Test case 3: Verify the cart persists across sessions.
  - Test case 4: Verify error handling when adding an out-of-stock item.
- Without NLP, a human tester would manually write these cases.
- With NLP, the system **automates the interpretation of the text and speeds up test preparation.**

Predictive Analytics in Testing

- **Predictive analytics** uses historical data, statistical models, and machine learning to forecast future outcomes in the testing process.
- It helps testers and managers make informed decisions about **where, what, and how** to test more effectively.
- **How Predictive analytics is used:**
  - **Defect Prediction:** Predicts which modules or features are likely to have defects based on past bugs, code changes, and complexity.
  - **Test Effort Estimation:** Estimates the amount of time and resources needed for testing based on previous project data.
  - **Test Case Prioritization:** Ranks test cases by likelihood of finding defects, helping focus on high-value tests first.
  - **Release Readiness Prediction:** Forecasts whether the software is stable enough for release based on defect trends and testing progress.

Predictive Analytics in Testing – *An Example*

- **Example Scenario:** Defect Prediction
- **Context:** Your team maintains a large web application. Historically, bugs are more frequent in certain modules after big changes.
- **How Predictive Analytics Helps:**
  - You collect historical data: Code churn (how often code changes) | Module complexity | Past defect density | Developer activity
  - You train a predictive model that identifies modules likely to have defects in the next release.
  - The model predicts that the "payment" and "checkout" modules have a high risk of defects.
- **Outcome:**
  - You prioritize test cases for those modules.
  - Allocate more testers and automation to those areas.
  - Prevent costly defects before release.

Clustering & Classification in Testing

- **Clustering** is an unsupervised learning technique used to group similar data points without predefined labels.
- **Classification** is a supervised learning technique that assigns labels to data based on learned patterns from labeled examples.
- Both are widely used in software testing to organize test data, streamline processes, and improve defect prediction.
- **How Clustering is used**:
  - Groups similar test cases or bug reports to identify duplicates, overlaps, or patterns.
  - Helps optimize test suites by removing redundant cases.
- **How Classification in Testing**:
  - Predicts outcomes like test case failure, defect severity, or module risk.
  - Automatically filters and routes bug reports (e.g., valid vs. invalid, critical vs. minor).

Clustering & Classification in Testing - Example

- **Example Scenario:** Bug Triage System
- **Clustering:**
  - You have thousands of bug reports in free-text format.
  - A clustering algorithm (like K-means or DBSCAN) groups similar bug descriptions.
  - You notice multiple clusters of bugs related to “login errors” and “payment failures.”
  - This helps in identifying duplicate reports and prioritizing high-impact issues.
- **Classification:**
  - You train a classifier (like Decision Tree or SVM) on historical bug data with features such as: Text of the report, Component affected, Steps to reproduce,
  - Labels include "High," "Medium," or "Low" severity.
  - New bug reports are automatically classified into severity levels to assist developers in prioritizing them.

Reinforcement Learning in Testing

- **Reinforcement Learning** is a type of machine learning where an agent learns to make decisions by interacting with an environment, receiving rewards or penalties based on the outcomes of its actions.
- In software testing, RL is used to **learn optimal testing strategies over time.**
- **How Reinforcement Learning is Used**:
  - Test case prioritization
  - Test path exploration (especially for UI or API testing)
  - Automated bug discovery
  - Dynamic test suite optimization
  - Self-healing test automation

Reinforcement Learning in Testing - Example

- **Example Scenario:** UI Testing with Reinforcement Learning
- **Context:** You are testing a web application's user interface. Manually exploring all possible user interactions is time-consuming.
- **How RL Works Here:**
  - **Agent**: The RL agent acts as an automated tester.
  - **Environment**: The web application's UI.
  - **States**: Current screens/pages and element states.
  - ***Actions***: Clicking buttons, filling forms, navigating pages.
  - ***Reward***: Positive reward for finding bugs or increasing coverage; penalty for dead ends or redundant paths.
- **Workflow:**
  - The agent starts on the homepage.
  - It chooses to click a button (e.g., "Login").
  - If the action leads to a new page or exposes a bug (e.g., error message), the agent gets a reward.
  - Over many test sessions, the agent learns which actions yield the most information or errors.
  - Eventually, it builds a test strategy that explores the application efficiently and effectively.

Anomaly Detection in Testing

- **Anomaly detection** involves identifying data or behavior that significantly deviates from what is expected or typical during testing.
- It helps uncover issues not directly covered by test cases - such as performance drops, unexpected errors, or subtle bugs.
- **How Anomaly Detection is Used:**
  - Detecting performance regressions
  - Identifying unstable modules
  - Spotting new, rare, or unexpected bugs
  - Monitoring system logs for unusual events

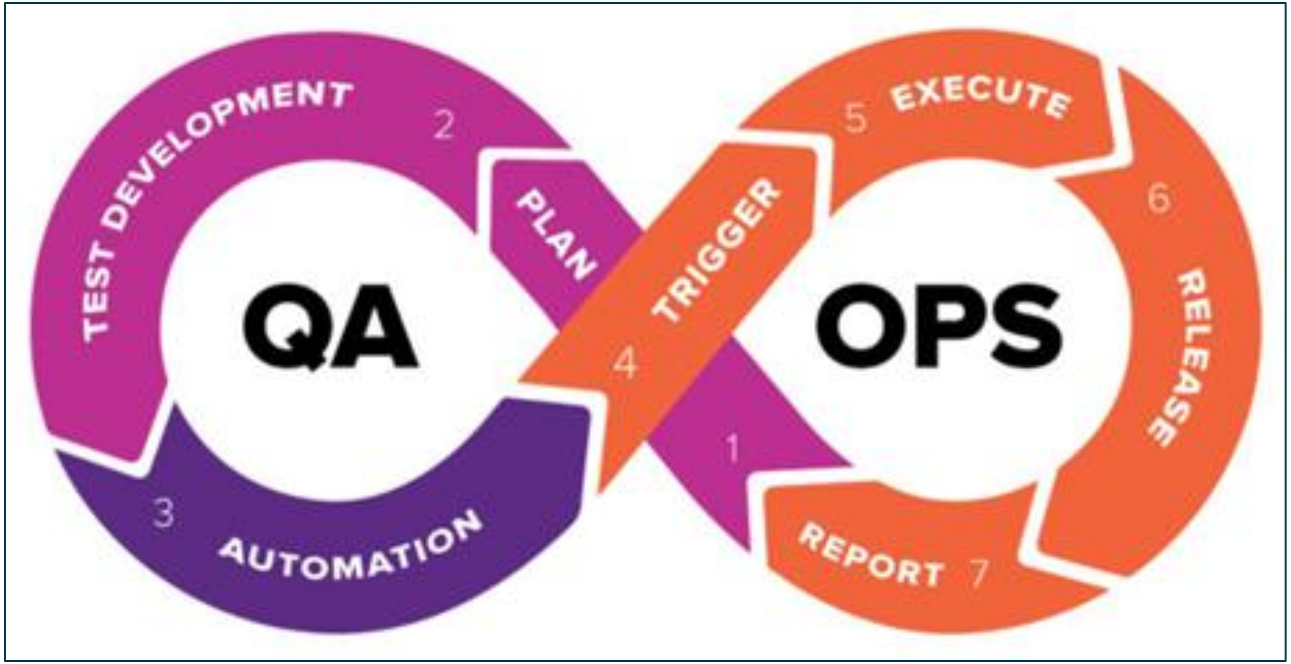
Anomaly Detection in Testing - Example

- **Context:** You are testing a web application, and you routinely run load tests to measure API response time under varying user loads.
- **Expected Behavior:** Under normal conditions, the API should respond in 100–200 milliseconds.
- **Observed Behavior During a Test Run:**
  - 90% of requests: ~150 ms (normal)
  - 10% of requests: 700 ms (unexpected spike)
- **How Anomaly Detection Helps:**
  - A statistical or machine learning-based anomaly detection model analyzes the response time data and flags the 700 ms results as anomalies, even though the test case itself did not "fail."

3. Rising significance of QAOps

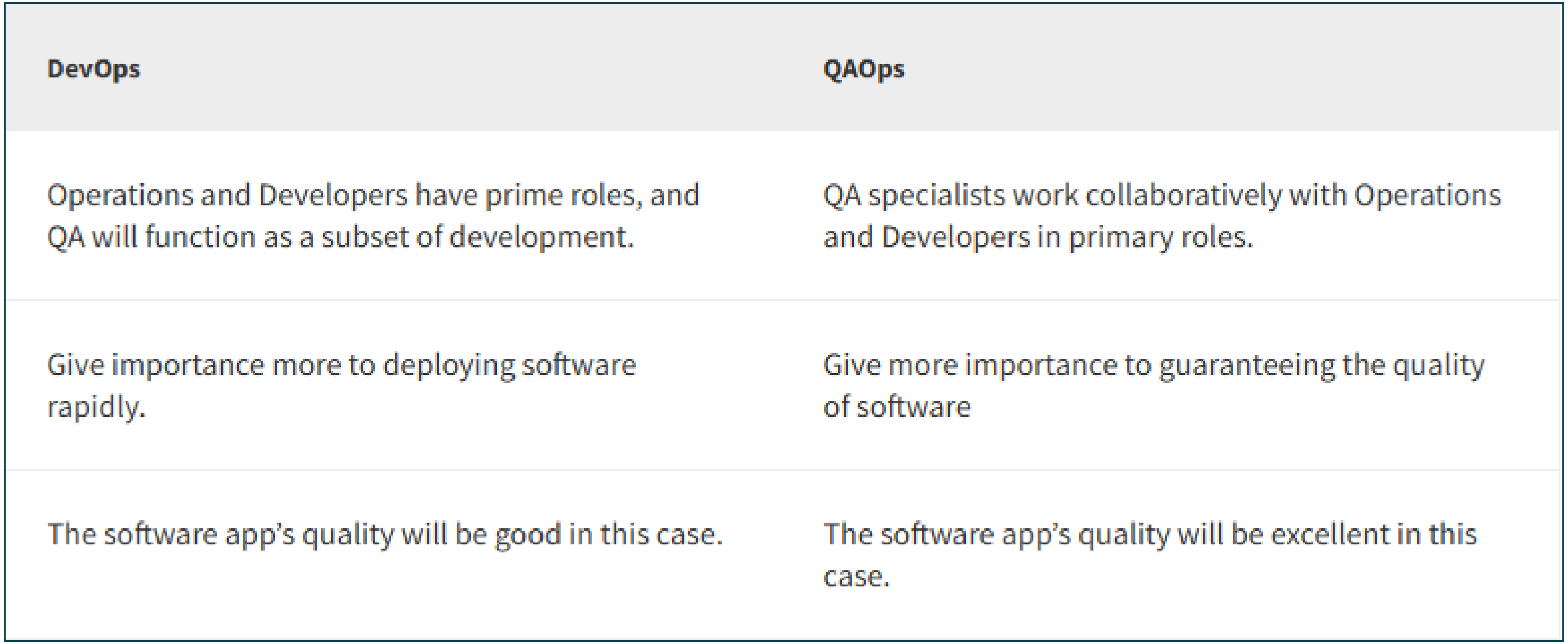
- QA is no longer just a separate testing phase
- It is **embedded continuously** throughout development, deployment, and delivery, using **automation, tools**, and **close collaboration** between QA, development, and operations teams.
- Key Ideas:
  - **Continuous** testing across the CI/CD pipeline.
  - **Automated test execution** after each code change or deployment.
  - **Shift-left testing –** QA works early in the development process.
  - **Monitoring in production** for ongoing quality (not just pre-release).

QAOps Cycle



QAOps **integrates** QA into the DevOps cycle

DevOps vs QAOps

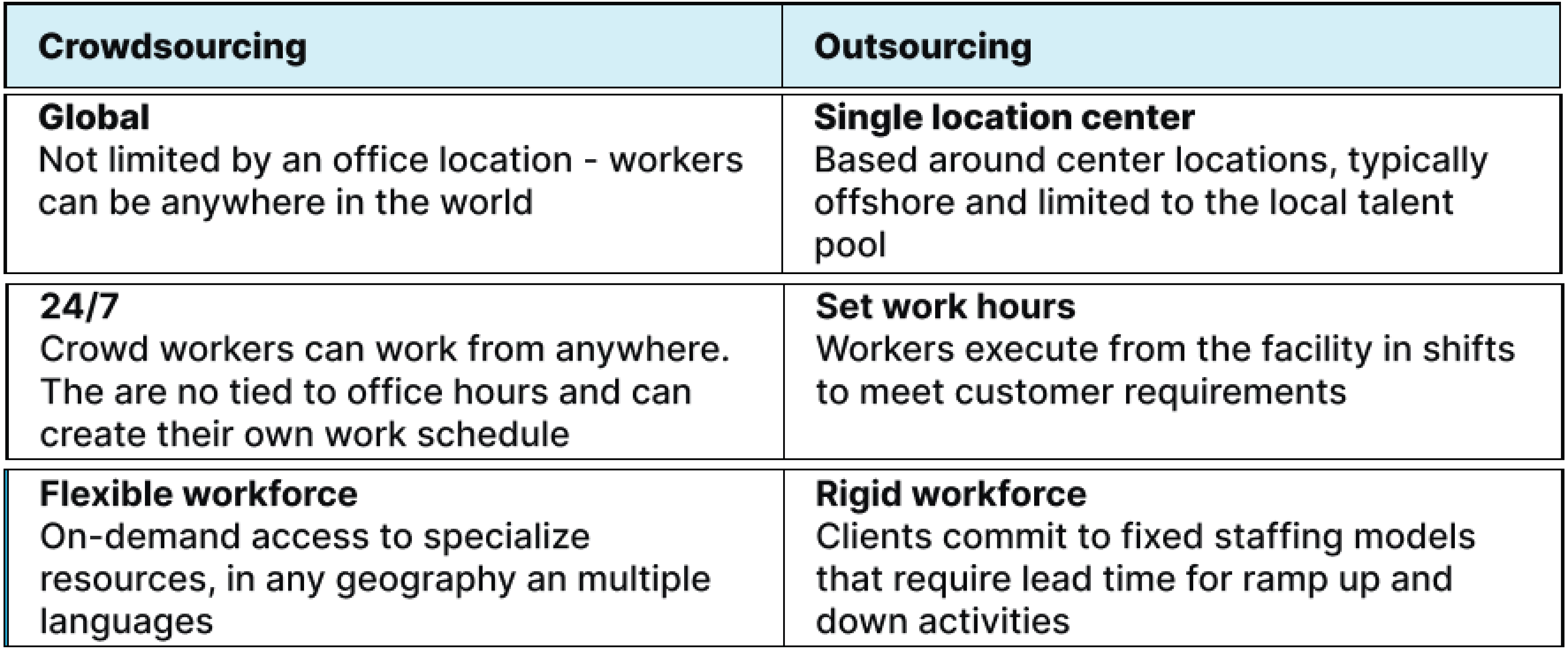


DevOps	QAOps
Operations and Developers have prime roles, and QA will function as a subset of development.	QA specialists work collaboratively with Operations and Developers in primary roles.
Give importance more to deploying software rapidly.	Give more importance to guaranteeing the quality of software
The software app’s quality will be good in this case.	The software app’s quality will be excellent in this case.

4. Crowdtesting

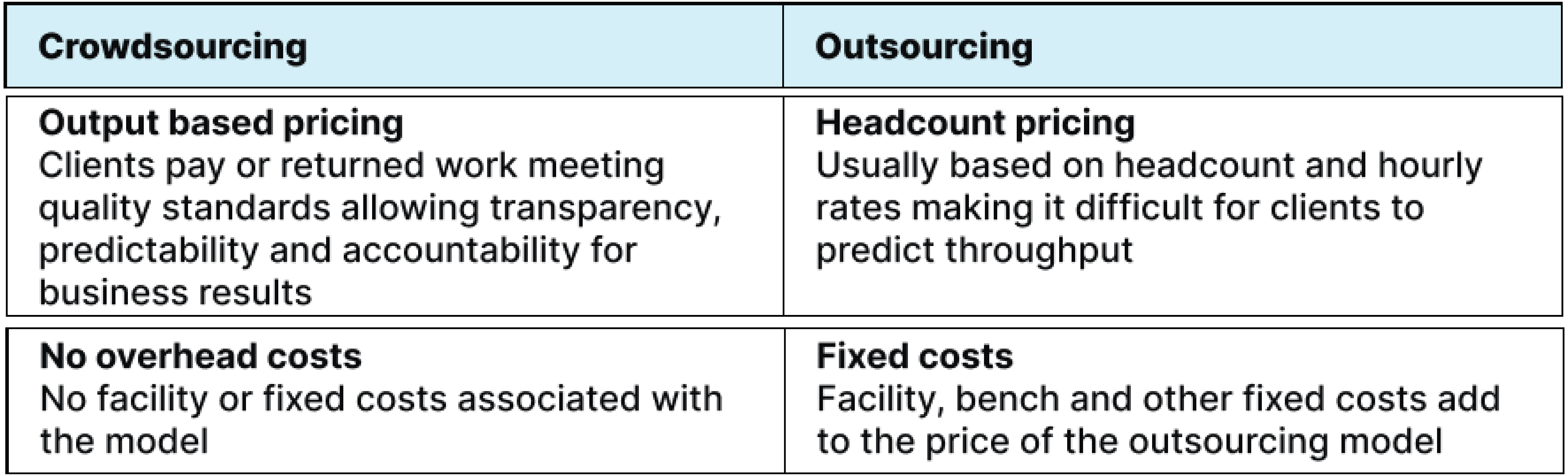
- A method that involves a large group of testers **who are not part of the company's internal QA team**
- Engage with these testers **through crowdsourcing platforms** (Amazon Mechanical Turk, Upwork, 99designs, etc.) where they can submit their software or applications for testing.
- As more organizations adopt this strategy, the global market for crowdtesting is **expected to expand.**

Crowdsourcing vs Outsourcing



Crowdsourcing	Outsourcing
Global Not limited by an office location - workers can be anywhere in the world	**Single location center** Based around center locations, typically offshore and limited to the local talent pool
24/7 Crowd workers can work from anywhere. The are no tied to office hours and can create their own work schedule	**Set work hours** Workers execute from the facility in shifts to meet customer requirements
Flexible workforce On-demand access to specialize resources, in any geography an multiple languages	**Rigid workforce** Clients commit to fixed staffing models that require lead time for ramp up and down activities

Crowdsourcing vs Outsourcing



Crowdsourcing	Outsourcing
Output based pricing Clients pay or returned work meeting quality standards allowing transparency, predictability and accountability for business results	**Headcount pricing** Usually based on headcount and hourly rates making it difficult for clients to predict throughput
No overhead costs No facility or fixed costs associated with the model	**Fixed costs** Facility, bench and other fixed costs add to the price of the outsourcing model

Benefits of Crowdtesting

- Scalable testing resources that are available as needed.
- More comprehensive test coverage.
- Quicker feedback loop with end users.
- The influx of specialized expertise and experience.

5. Evolution of Test Automation

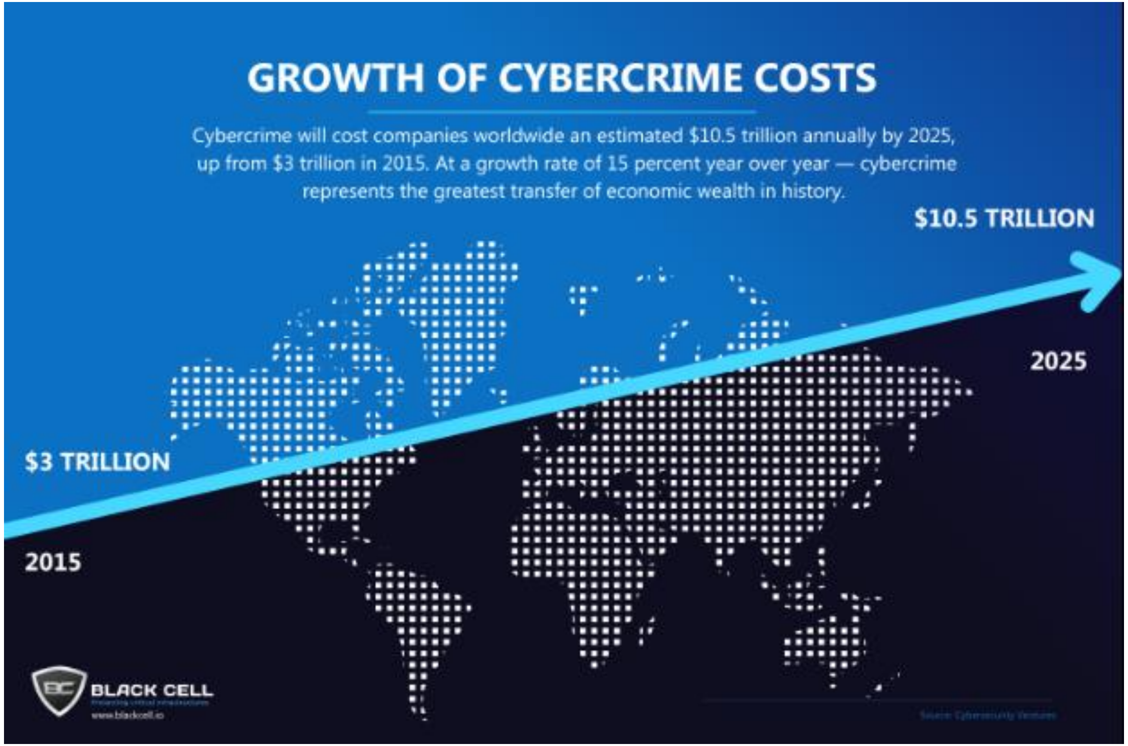
- Revise ‘Test Automation’ lecture…
  - Introduction to Test Automation
  - Manual Vs Automated Testing
  - Principles of Test Automation
  - Test Automation Lifecycle
  - Test Automation Frameworks
  - Challenges & Myths in Test Automation

6. Enhanced focus on Security Testing

- **Cybersecurity threats and data breach** incidents surged recently
- Integrating security from the **initial stages of product design and development** is now essential.
- Emerging field of **DevSecOps**
- Emphasizes the importance of various security testing methodologies:

  Vulnerability scanning | Penetration testing | API testing | Web application security testing.

Cost of Cybercrime



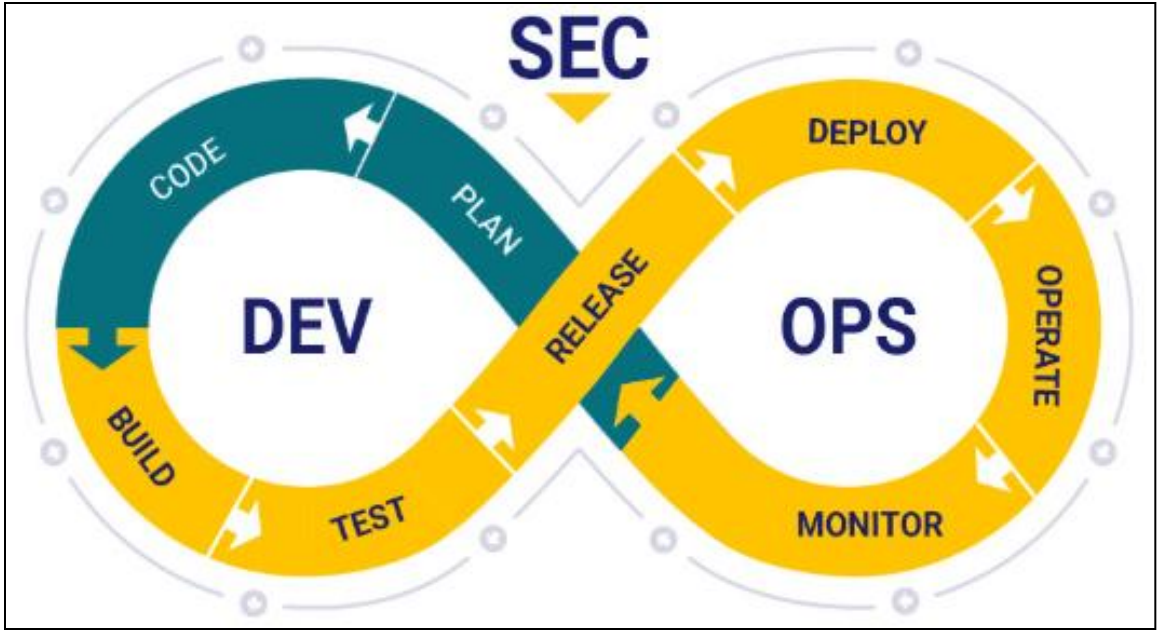
GROWTH OF CYBERCRIME COSTS

Cybercrime will cost companies worldwide an estimated $10.5 trillion annually by 2025, up from $3 trillion in 2015. At a growth rate of 15 percent year over year — cybercrime represents the greatest transfer of economic wealth in history.

Source: Cybersecurity Ventures

The cost of cybercrime is predicted to grow exponentially, with a **70% increase by 2028.** This means the annual cost will jump from $8.15 trillion in 2024 to **$13.82 trillion by 2028**

DevSecOps

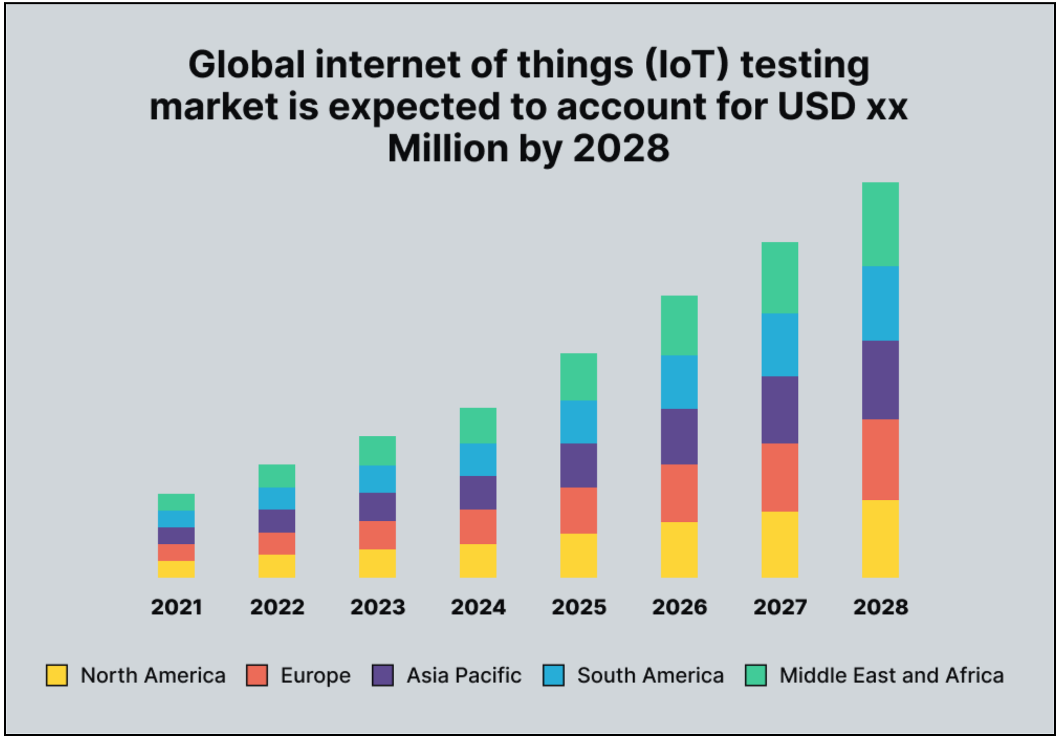


- DevSecOps is a methodology to provide security to application and infrastructure based on the principles of DevOps.
- This approach makes sure that the application is less vulnerable and ready for user's use.
- An automated process, and security checks started from the beginning of the application's pipelines.

7. IoT (Internet of Things) Testing

- Significant software testing trend focusing on IoT Instruments':
  - Security
  - Data integrity
  - Performance
  - Scalability
  - Compatibility.
- Is crucial for enhancing system productivity by preventing unexpected glitches.
- provides greater control over devices and helps improve network and device efficiency, accessibility, and usage for the users

IoT Testing Market



Global internet of things (IoT) testing market is expected to account for USD xx Million by 2028

Significant growth in IoT Testing market reflects:

- The **increasing reliance** on IoT devices
- the need for effective testing to ensure their **functionality and security.**

8. Mobile Test Automation

- Development of mobile applications is **rapidly growing,** hence mobile test automation will greatly help
- Mobile test automation involves **using software tools and scripts** to **automatically test** mobile applications across **various devices and platforms.**
- Integrating **cloud-based mobile labs** and test automation tools represents a promising development, potentially elevating mobile test automation to new heights.

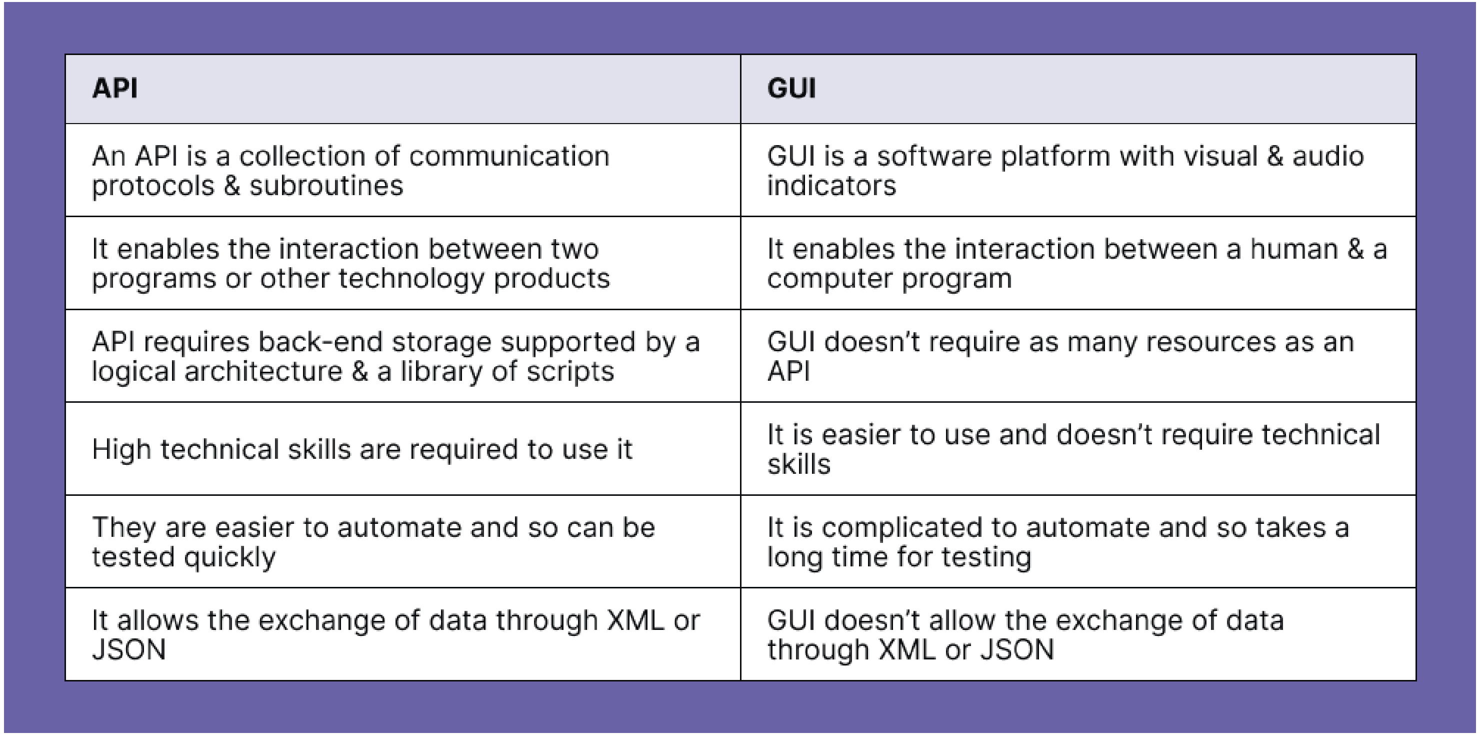
Cloud-based Mobile Labs

- A remote testing environment where mobile applications can be tested on various **real devices or emulators** hosted in the cloud.
- This allows testers to access and test apps on different operating systems, device models, and screen sizes **without the need for a physical on-site lab**
- **Key Features:** Remote Access | Real Devices | Virtual Devices | Scalability | Cost-Effectiveness | Flexibility | Integration
- **Examples of Cloud-Based Mobile Labs:** Sauce Labs | BrowserStack | LambdaTest | AWS Device Farm | Perfecto.io | Kobiton

9. Enhanced API testing and Automation

- Rise of microservices architectures has significantly increased the number of **Application Programming Interfaces (APIs)**
- Hence API Test Automation is way to go (***"Make frequent cases faster"***)
- Increasingly becoming the **standard practice** as API-driven development more relevant than ever
- API test automation is the solution for **achieving higher efficiency**, enabling more tests to be performed in a shorter time

API Testing Vs GUI Testing



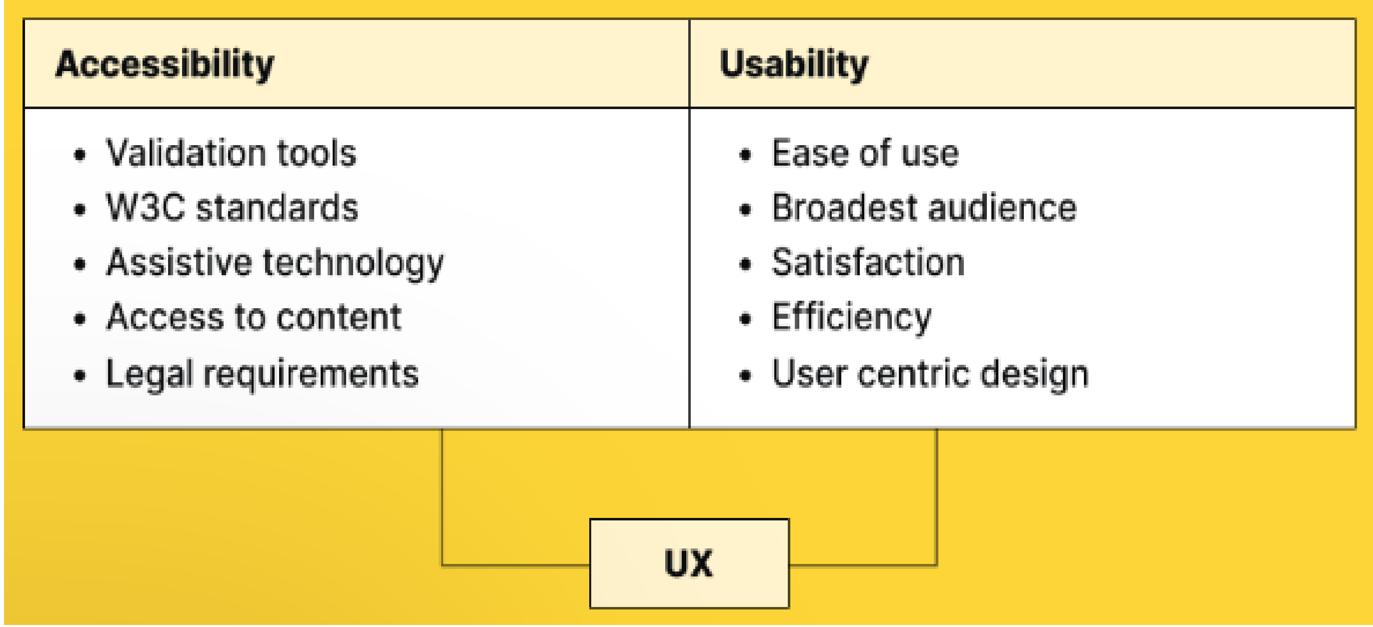
API	GUI
An API is a collection of communication protocols & subroutines	GUI is a software platform with visual & audio indicators
It enables the interaction between two programs or other technology products	It enables the interaction between a human & a computer program
API requires back-end storage supported by a logical architecture & a library of scripts	GUI doesn't require as many resources as an API
High technical skills are required to use it	It is easier to use and doesn't require technical skills
They are easier to automate and so can be tested quickly	It is complicated to automate and so takes a long time for testing
It allows the exchange of data through XML or JSON	GUI doesn't allow the exchange of data through XML or JSON

API testing is a more efficient alternative to extensive GUI testing in Agile or DevOps environments, where speed is crucial.

10. Focus on Accessibility Testing

- This is an era of stringent **web and mobile accessibility regulations**
- Compliance through accessibility testing has become a critical component of software development.
- Accessibility standards: ***Americans with Disabilities Act (ADA), Section 508, and the Web Content Accessibility Guidelines (WCAG) 2.1.***
- Crowdtesting offers valuable perspectives from users with disabilities.
- For global applications, refining accessibility testing practices is essential to **bridge compliance gaps** and cater to a **diverse user base**

Focus Factors on Accessibility / Usability

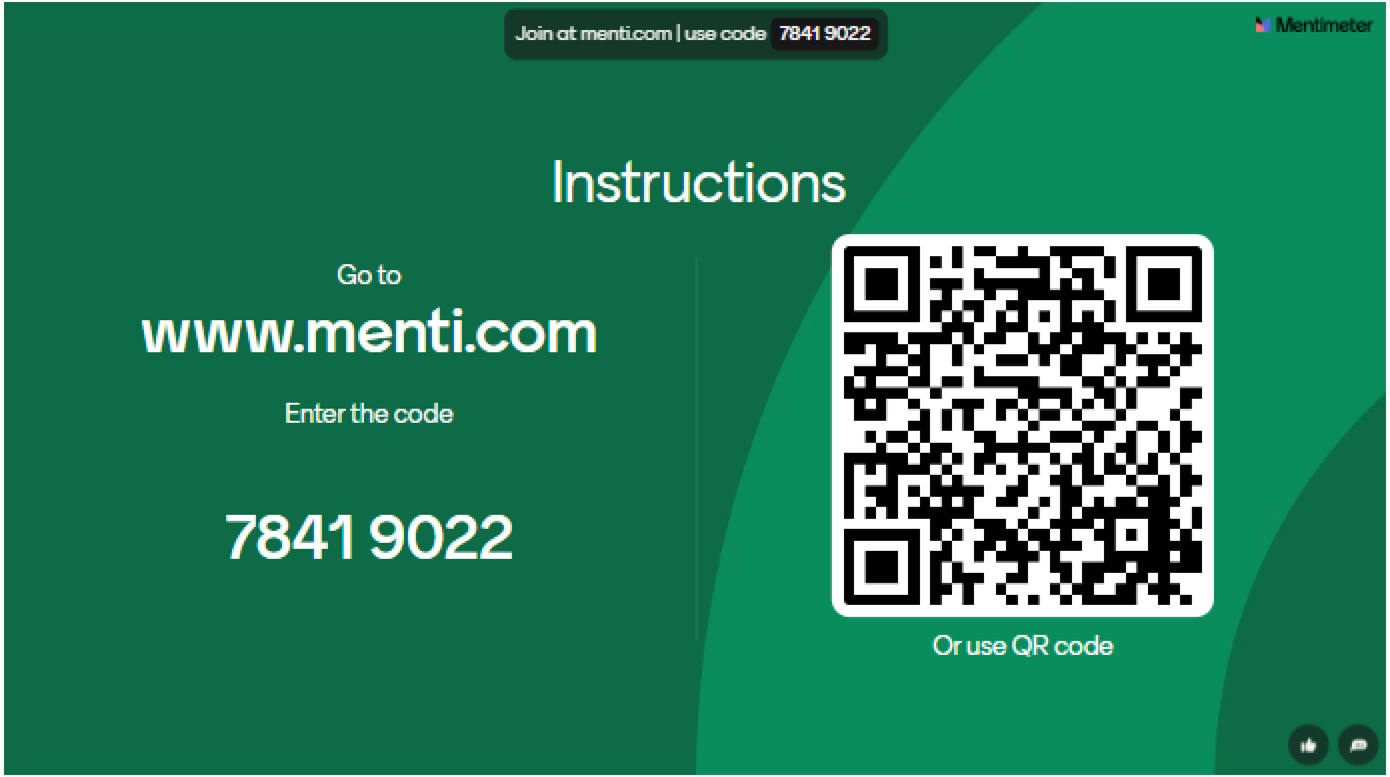


Accessibility	Usability
- Validation tools - W3C standards - Assistive technology - Access to content - Legal requirements	- Ease of use - Broadest audience - Satisfaction - Efficiency - User centric design

UX

Incorporating automation and AI in testing can efficiently handle the repetitive aspects (**screen readers, magnifiers, and captions).**

Quiz Time



Join at menti.com | use code 7841 9022

Mentimeter

Instructions

Go to

www.menti.com

Enter the code

7841 9022

Or use QR code

Thank You

Q & A

suresh.n@sliit.lk | 755841849