CAREER PATH • SOFTWARE TESTING

# Software Testing Career Path

**Build release evidence — not just test-case screenshots.**

Manual testing • Agile • Jira
SQL • API testing • Performance
Portfolio evidence for QA interviews

Requirement → Test → Defect → Evidence → Release

ITLearn360

# What You Will Be Able to Explain

Outcome-focused, not tool-name memorization

**01**

**Role lens**

**02**

**Workflow lens**

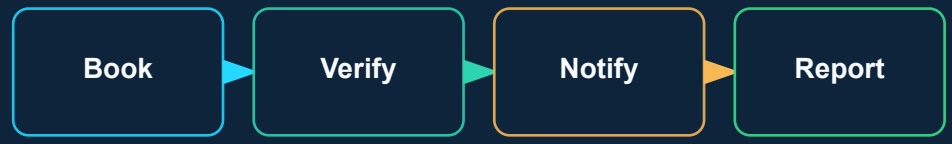
- How software testing fits into real delivery work
- How a tester turns requirements into test evidence
- How manual, API, SQL and performance checks connect
- How to present a testing portfolio in interviews

# Workplace Scenario: Lakeside Health Portal

One fictional release connects the entire presentation

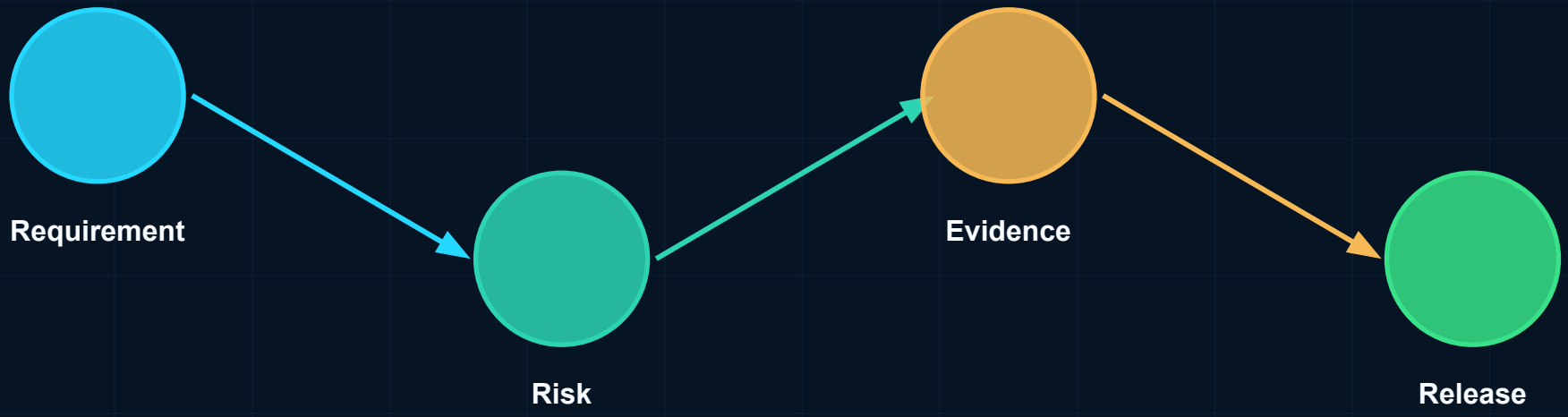
## Lakeside Health Portal

- Patients book appointments, receive reminders and view billing status
- A new release changes login, provider search and scheduling
- QA must test user flow, API behavior, database updates and performance
- The release decision depends on evidence, not assumptions



# Role Reality: Tester as Quality Investigator

Testing is not random clicking; it is structured risk investigation




- Clarify expectations before execution
- Ask what could fail and who would be affected
- Record evidence that developers and managers can act on
- Protect user trust before the release reaches production

# Start With Requirements, Not Test Cases

A tester first understands the decision behind the feature

- Identify the business rule behind each user story
- Ask acceptance-criteria questions early
- Separate happy path, negative path and boundary conditions
- Mark unclear requirements before they become defects

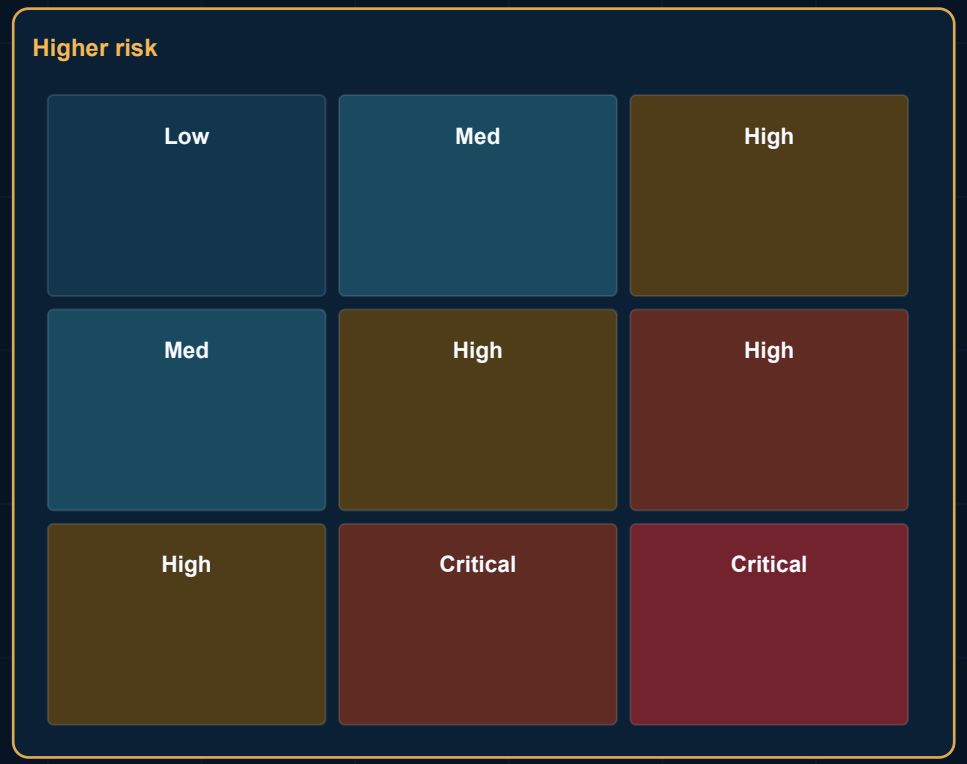
**Scenario question**

If a patient edits appointment time twice, what should happen to the confirmation message and database record?

# Build a Risk-Based Test Strategy

Not every feature receives the same testing depth

- Critical patient workflows receive deeper testing
- Low-impact display changes may need lighter checks
- Risk comes from impact, likelihood and recent change
- Strategy explains why the team tests in a specific order




## High-risk first

Login, provider search and appointment booking receive deeper QA because failure directly affects user access and scheduling.

# Agile and Jira: Where QA Work Becomes Visible

Testing evidence must move with the sprint


- QA participates during refinement, not only after development
- Test tasks connect to user stories and acceptance criteria
- Defects include steps, data, screenshots and environment details
- Sprint reporting should separate blocked, failed and passed work

# Manual Test Design: Think Like the User and the System

Manual testing is deliberate exploration with repeatable evidence


- Design tests around real user journeys
- Use equivalence classes and boundary values where useful
- Include accessibility, usability and data-validation checks
- Keep exploratory notes separate from scripted execution

**Test idea**

A patient searches by specialty, selects a provider, changes time, and confirms appointment. QA observes both the UI behavior and the backend scheduling result.

# Defect Reports That Developers Can Fix

A good bug report reduces investigation time

## Poor report

*“Appointment is broken.”*

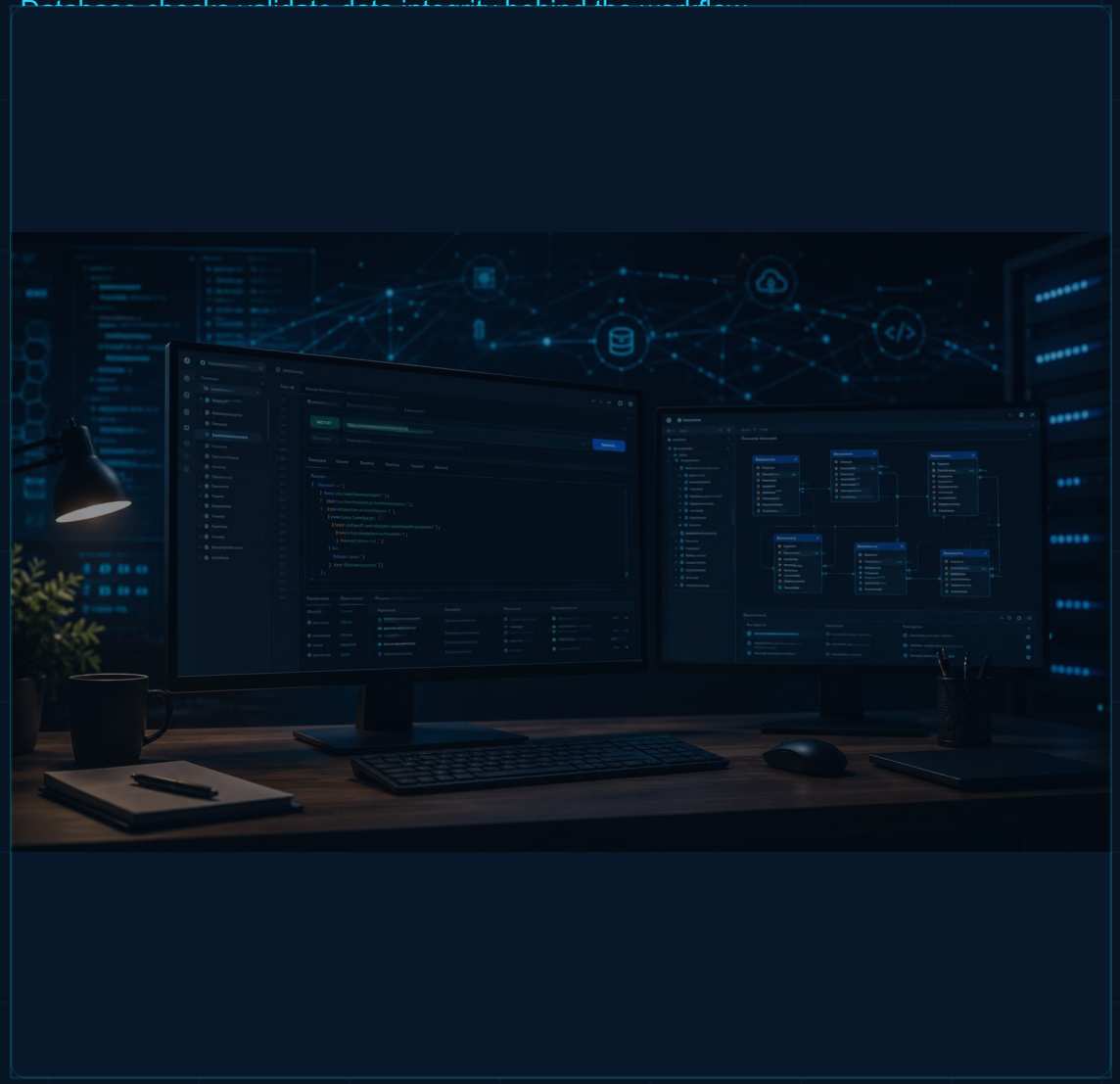
No steps, no data, no user impact, no evidence.

## Strong report

- State the user impact before technical details
- Provide steps, actual result and expected result
- Attach evidence: data, screenshots, logs or request IDs
- Classify severity by risk, not emotion

# SQL Testing: Verify What the UI Cannot Show

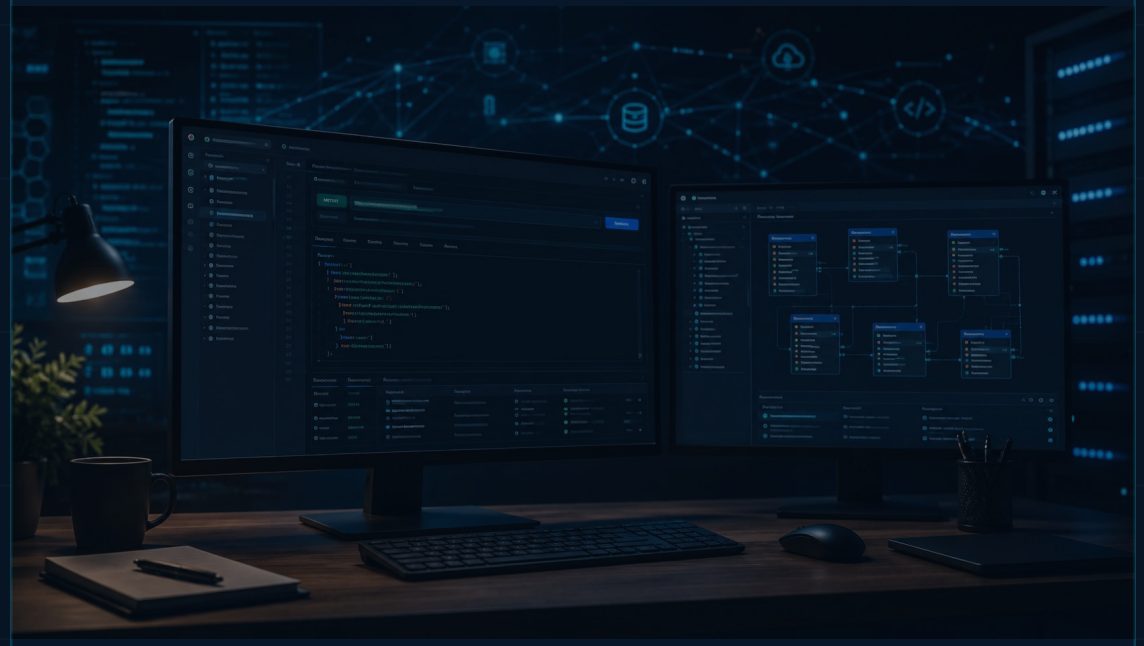
Database checks validate data integrity behind the workflow



- Confirm appointment records are created correctly
- Verify user, provider and scheduling data stay consistent
- Check update, cancellation and audit fields
- Use read-only queries and synthetic test data

# API Testing With Postman

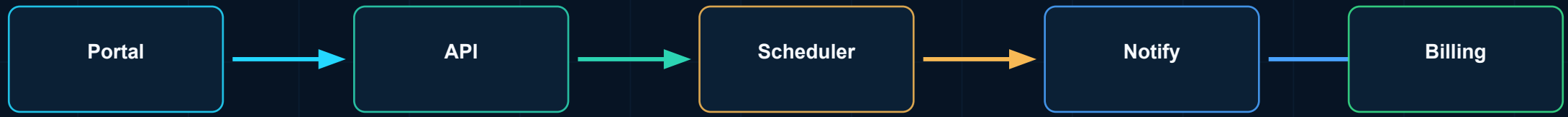
Modern QA validates the service layer directly



- Test authentication, status codes and response schemas
- Check positive, negative and boundary payloads
- Validate business rules before UI integration is complete
- Save collections as portfolio evidence

# Integration and Web Services Testing

Quality issues often appear between systems




- Appointment, notification and billing systems exchange data
- SOAP or legacy services may still support enterprise workflows
- Test contracts, error handling and retry behavior
- Document what system owns each failure

# Performance Testing With JMeter

A feature can be correct but still fail under load.

- Define the workload before running a test
- Track response time, throughput and errors
- Watch database, API and infrastructure signals together
- Use results to guide capacity and release decisions

# Regression, Re-Testing and Release Readiness

The final decision depends on what changed and what was proven


- Re-test fixed defects with the original failure data
- Run regression around impacted workflows
- Document open risks and known limitations
- Support go/no-go decisions with evidence

**Release question**

What evidence proves the appointment workflow is safe enough to release today?

# Tools Connected to QA Tasks

Tools matter only when the testing task is clear

### Jira

Defects + sprint visibility

### Postman

API requests + assertions

### SQL

Data validation

### JMeter

Load + performance evidence

### Sheets

Test data + reporting

**Each tool must tie to a task and a decision.**

# Common Mistakes to Avoid

Small habits can weaken both testing and interview credibility

## Weak approach

- Writing test cases without understanding risk
- Logging vague defects without useful evidence
- Ignoring API and database behavior

## Better approach

- Treating performance as a last-minute checkbox
- Claiming coverage without explaining what was actually tested

# Portfolio Lab: Test the Lakeside Release

Build one connected case instead of random screenshots

- Test plan
- Execution log
- Defect reports
- API + SQL evidence
- Performance summary

**Portfolio proof = decisions + evidence + explanation.**

# ISTQB and Learning Path

Certification helps when it supports practical testing skill

- Use ISTQB CTFL v4.0 for core testing vocabulary and principles
- Practice Agile, Jira, SQL, Postman and JMeter through one project
- Do not treat certification as a job guarantee
- Turn every module into an interview-ready artifact

## Learning path


Then connect the artifacts into a single QA story.

# U.S. Job-Market Reality Check

Use market data as directional context, not a promise

**10%**

BLS projected growth for Software QA Analysts and Testers, 2024–2034

**14K**

Approx. average annual openings in the same BLS line item

- Job boards change daily and include duplicates or adjacent titles
- Employers still look for communication, documentation and evidence

No salary figures included. Update job-board counts again on publish date.

# Interview Story and Practical Action Plan

Explain the work like a tester, not a tool collector

1. **Pick one application workflow**
2. **Create manual + API + SQL evidence**
3. **Add performance summary**
4. **Write release-readiness story**

- Pick one application workflow and write the test strategy
- Create manual, API, SQL and performance evidence
- Write two strong defect reports and one release summary
- Practice explaining tradeoffs, risk and user impact
- Close with what you would improve next

**ITLearn360 • Speed • Clarity • Practical IT Learning**