Interviewer Assistance Questions for John

These questions assess John's problem-solving skills based on his strengths (Puppeteer, Swift) and areas for improvement (C++, CI/CD, Big Data Testing).

---

1. Puppeteer & Automation - Handling Captcha Challenges

Scenario:

John is automating a web application using Puppeteer. Some login tests fail because the site occasionally displays a CAPTCHA challenge, preventing automated sign-ins.

Question:

How would you handle CAPTCHA challenges in Puppeteer test automation?

Expected Answer:

- **Request CAPTCHA whitelisting for automation accounts if possible.**
- **Use browser session persistence (cookies/storage) to avoid repeated logins.**
- **Implement manual intervention hooks to allow human input when necessary.**
- **Leverage CAPTCHA-solving APIs (e.g., 2Captcha, Anti-Captcha) if permitted.**
- **Explore headless browser fingerprinting techniques to minimize bot detection.**
- **Consider mocking API responses for login validation in non-production environments.**

---

2. Swift & Mobile Testing - Debugging Crashes in iOS App

Scenario:

John is testing an iOS application written in Swift. After a recent update, the app crashes on launch for some users but runs fine on his test device.

Question:

What steps would you take to diagnose and fix the crash issue?

Expected Answer:

- **Check crash logs in Xcode (Console & Devices) to identify error messages.**
- **Use TestFlight logs to gather data from affected users.**
- **Verify dependency versions (CocoaPods/Swift Package Manager) for compatibility.**
- **Test with different iOS versions and device models to find inconsistencies.**
- **Debug using breakpoints and memory profiling tools (Instruments in Xcode).**
- **If the crash is user-specific, inspect local storage, permissions, and network conditions.**

---

3. C++ & Performance Optimization - Memory Leak Issue

Scenario:

John is working on a C++ backend service that processes real-time data. Over time, the service's memory usage keeps increasing, leading to system slowdowns.

Question:

How would you detect and fix memory leaks in C++ applications?

Expected Answer:

- **Use Valgrind or AddressSanitizer to detect memory leaks.**
- **Review dynamic memory allocations (`new` / `delete`) to ensure proper deallocation.**

- **Implement smart pointers (`std::unique_ptr`, `std::shared_ptr`) to manage memory automatically.**
- **Use RAII (Resource Acquisition Is Initialization) to ensure cleanup.**
- **Monitor heap usage with tools like gperftools to detect growing allocations.**
- **Optimize data structures to minimize unnecessary memory usage.**

---

4. CI/CD Pipelines - Failing Build After Code Merge

Scenario:

John’s team uses Azure DevOps for CI/CD. After merging a new feature branch, the build fails in the pipeline but passes locally.

Question:

How would you diagnose and resolve this pipeline failure?

Expected Answer:

- **Review pipeline logs to identify the exact failure point.**
- **Ensure all dependencies and environment variables match between local and CI.**
- **Check for missing files (e.g., config files ignored by `.gitignore`).**
- **Validate permissions for build agents (e.g., access to private repositories or cloud resources).**
- **Run the build using Docker or a CI sandbox locally to replicate the failure.**
- **If the failure is due to flaky tests, implement retry mechanisms and logging.**

---

5. Big Data Testing - Ensuring Data Accuracy in a Hadoop Pipeline

Scenario:

John’s company processes large datasets in Hadoop. The QA team finds that some processed data is incomplete or incorrect after transformation jobs run.

Question:

How would you validate data accuracy and integrity in big data testing?

Expected Answer:

- **Implement data validation scripts to compare input vs. output records.**
- **Use checksums or hashing to detect data corruption.**
- **Perform row count and schema validation to catch missing records.**
- **Set up unit tests for transformation logic in Spark or Hive queries.**
- **Monitor job execution logs for failures in data processing steps.**
- **If using distributed processing, verify data shuffling and partitioning logic.**