1. **Test Automation Solution**

* **High-level test automation stragety:**
  + Aim for high automation coverage acress Web, Mobile, API: Define P1, P2, P3, P4 test cases > Try to automate P1 test cases first for critical user journeys and high-risk areas coverage then aim for higher coverage by implement P2 > P4 automation test cases. Avoid P3, P4 test cases that brings low ROI, high maintenance fee.
  + Implement automation test script as a quality gate to release code to production (starting from P1 script), integrate with Slack or Teams to get fast testing result and involves the developer in the debugging process
  + Implement reusable tests: design tests to be reusable, easy to maintain (POM would be one solution, with Cypress we can implement Application Actions as their most recommended design pattern)
  + Shift-left testing: Using component testing and unit testing to involve early in development process, where defects are cheaper in price and easier to resolve.
* **Recommendation for framework, tools and technologies**: As a Senior QA Automation engineer who has prior experiences working with various e2e automation testing tool I would recommend Cypress framework for automating **component**, **API** and **integration testing**. Compared with Selenium, which has lower performance due to webdriver (initiate webdriver will be slower than being triggered from same loop as the application). Moreover, the experience of members within the project should be put into consideration. I have more than 4 years working with Cypress with a detailed mentoring framework created, if there are some people with same or higher level who prefers other Framework over Cypress, I would definitely give it a chance, especially with Playwright where Cross-browser testing is by defailt, advanced parallel algorithm with growing community.
* With Mobile automation, most of people would definitely recommend Appium as a strongest framework for native mobile applications with strongest community. However, as had been experienced in current project, Appium is really slow in term of performance and similar to Selenium, ỉt requires more manual handling of waits and synchronization. As I have used Detox for testing **React Native** project, the speed is way much faster because it tests directly within the app’s process. More over it is native support for react native, using Javascript, have similar automatic synchronization as Cypress for web application. However, one drawback of using Detox is that the framework currently not supporting IOS emulator or real device. The two framework can all be put into consideration for our own purpose. For me with trading app, performance need to be put into consideration more than mobile animation so Detox would be better fit despite of its drawback. Another alternative is Maestro which is a record and playback framework if the team is under heavy work load and the environment is fast-pace.
* Postman would be a good alternative to Cypress and Newman can be used to automate API tests.
* With Backend Services, I would recommend Jmeter, Gatling or K6 for Performance measuring. 3 of the tools when applying well load strageties can help us analyze our system in different conditions perfectly. I would like to suggest an APM tool like App insight or Datadog for better monitoring (personally I’m using Datadog to create metrics for better monitoring and receiving alerts via Teams or Outlook mail if Widgets reaching certain limit ( self pre-defined)
* Reporting: With Cypress we can use mochaawesome as report output, with Selenium we can use Allure or Cucumber dashboard to visualize test results and trends. Me personally prefer Browserstack/Cypress.io dashboard for gaining better overview over multiple run times.

**Recommendation for CI/CD Integration:**

Pipeline integration: Integrate automated test into CI/CD pipelines using tools like, Azure CI, Jenkins, Gitlab. Set up cron in yaml file for nightly run, add myself as code reviewer to control code commits, pull requests, test folder architecture

Reporting: depends on frameworks, using report tools I mentioned above

**Maintanability and Scalability:**

* Code reviews and best practices: Implement code reviews for test scripts, naming convention, using framework best practices (design pattern, locator query, explicit wait, retries, exception handling, debug, etc…)
* Version control: Use git for version control (ensure change can be tracked and reverted if needed)
* Regular refactoring: regular enhance the source code for better maintability, improve debug time by implement third party packages.

**2. AI Integration in QA & Testing:**

I would like to implement AI to write unit test for automation testing script for better coverage, remove negative postive cases, use AI to prioritize test cases based on historical data and risk assessment (cypress.io), implement AI to predict potential defects when new code implemented (similar to a statistic analysis code smell)

Self-healing tests: Honestly I haven’t used any tool with self-healing capability but researching shows two tools that stands out: Test.ai and Applitools that automatically update tests when UI changes

Test coverage optimization: I have experience using Sonarqube as a quality gate. TestCraft should be a good alternative

Exploratory testing: Xray exploratory app would be a good solution or any record and playback with LLMs

**3. Performance & Security Assurance:**

**Load** **testing**: Use K6, Jmeter, Gatling to simulate real-world load conditions and monitor the performance metric (avg response time, cpu usage, throughput, error rate (95%), latency, network I/O, Disk I/O

**Monitoring**: Like I mentioned above, we should use Datadog, Application Insight for better monitoring

**Strategies and tools for performance testing:**

**Stress** **testing**: Collect avg user request in Golden hour using GA, then come up with performance strategy:

* Stress testing: High slope, ramp-up fast with high load to determine system’s breaking point and ensure it can handle peak load (peak load = avg user request in Golden hour x 1.5) within 1 hour
* Continuous monitoring: Set up a plan for continuos performance monitoring and implement alerts to warn the team if any anomalies in PROD (Datadog)

**Security testing:**

* Conduct regular audits and penetration testing to identify and mitigate vulnerabilities
* Ensure all team members are trained on security on security best practices for trading platform

**4. Efficient shift-left testing:**

Early involvement of QA: Involve QA in the requirements and design phase, becomes a platform consultant in the early phase to identify technical blockage, weak point in the design when integrate with other features, security or any quality risks.

Automated unit testing: Ensure unit test is written, should apply AI to achive higher coverage and removes negative positive test cases.

Pair programming: facilitate knowledge sharing and catch defects early by understanding individual developer behavior

Cross-functional teams: This definition just being introduced to my organization shortly. The reason why this team brings a lot of benefit is that team will have wide range of knowledge acress domain and can be useful for detect defects earlier in development phase.

Static Code Analysis: SonarQube would be a good solution

CI/CD integration: Create pipeline in early stage help unit tests, static code analysis, non-functional test comes in earlier