QA Bootcamp Automated Testing Report

Application Under Test: GoQuant's internal testing platform
Test Period: October 22–29, 2025 (1 week)
Tester: Neha Shanbhag
Testing Framework: Playwright with TypeScript
Browsers Tested: Chromium 118.0, Firefox 119.0, WebKit 17.0

Executive Summary

Overview

This report presents a comprehensive testing evaluation of a financial trading platform “GoQuant's internal testing platform” using Playwright as the primary testing framework. The assessment covers functional, UI/UX, accessibility, and edge case testing across critical user journeys including authentication, trading operations, order placement and management, setting preferences and market data visualization. The platform exhibits moderate to high instability with critical gaps in error handling, data persistence, real-time communication, and validation layers. The system shows symptoms of architectural fragility across frontend-backend integration, particularly in UDP-based real-time features and state management.

During the testing phase, a total of 75 critical issues were identified under the Product Defects category. These issues represent actual application failures with a high impact on functionality and user operations. The most common root causes for these product defects include authentication failures, trading algorithm execution errors, account management API malfunctions, order placement and cancellation issues, UI element timeout errors, and data inconsistencies observed in various system tables. In addition, 7 high-priority issues were identified under the Test Defects category. They have a medium impact on the overall testing process. The likely root causes of these test defects include flaky test cases, incorrect timeout configurations, synchronization challenges, inconsistencies in test data, and environment-specific failures that affected the reliability of automated test execution.

Firstly, application instability appears to be a significant contributor. The test environment (https://test1.gotrade.goquant.io/) exhibited backend issues, including intermittent API endpoint failures and occasional database connectivity disruptions. These factors collectively led to inconsistent test outcomes and a high number of system-level errors. Timeout-related issues played a major role. The total execution time of approximately 2 hours and 50 minutes indicates frequent delays and long waits, often exceeding configured thresholds (e.g., 180,000 ms). These symptoms suggest underlying performance bottlenecks and network idle conditions that were not being properly met during execution. locator issues were identified within the UI automation scripts. Dynamic elements and changing DOM structures caused XPath locators to become stale, while elements were frequently not found within the allotted timeout periods. These problems led to repeated test failures, particularly in UI-heavy workflows. The observed failure rate is 65.79%.

Overall Platform Stability Assessment

Stability Rating: 4.2/10

The platform demonstrates solid core functionality but exhibits concerning security vulnerabilities in authentication flows and data validation gaps in trading operations. The real time communication has Pervasive failures, no redundancy, single point of failure. Data loss on restart, failed DB writes, no transaction integrity. Input Validation has missing validation layers, accepts invalid trades. Error handling results in generic errors, silent failures, poor user feedback. State management has state loss on navigation, binding issues. UI/UX responsiveness gives timeout issues, blank charts, component communication gaps. The platform exhibits characteristics of not production-ready for mission-critical trading operations. High risk of financial loss or data corruption. User trust erosion due to unreliable operations.Support burden from poor error messaging.

Recommendation: Prioritize immediate remediation of critical issues related to order placement before production deployment. Implement comprehensive validation by adding client-side validation for all form inputs. Enforce backend validation with clear error responses. Fix data persistence by auditing database write operations for missing commits and implement proper transaction management. Enhance error handling by mapping specific error codes to user-friendly messages and replace generic "UDP Server" errors with root cause details. Testing coverage should include unit tests for validation logic. Integration tests for API contracts. End-to-end tests for critical user flows. Chaos engineering for failure scenarios. Document error codes and troubleshooting steps by creating runbooks for common failure scenarios and establish coding standards for error handling.

Testing Methodology

Testing Approach

I employed a risk-based testing strategy that prioritizes high-impact financial operations and security-critical flows, targeting the 38 documented manual defects across the platform. This approach ensures critical business functions are validated first while systematically addressing moderate and low-risk areas. The automation test suit contains 114 testcases across multiple functional areas. Financial Loss / Data Integrity had Revenue loss, regulatory risk as business impact. For Instance Invalid inputs trade, Canceled orders In-Progress, Account deletion on restart.Core Trading Functionality had User abandonment, trust erosion as business impact. For instance Working Order visibility, Order History, Transfer failures, Fill calculations. Analytics & Reporting had Poor UX, operational inefficiency related to Post Trade Analytics, Billing exports, Chart visibility. UI/UX Enhancements caused minor inconvinience like Dropdown navigation, Date Picker.

The approach combined:

Behavior-Driven Development (BDD) principles for readable test scenarios ensures test scenarios are written in business-readable language making them accessible to stakeholders. Tests are organized around user workflows and business scenarios rather than technical implementations: Login scenarios (valid, invalid credentials, empty fields). Complete trading workflows (account setup, trade execution, order verification). Navigation flows (tab switching, keyboard shortcuts).

Page Object Model (POM) pattern for maintainable test architecture. Test logic separated from page interaction logic, UI changes required updates in single location, TypeScript interfaces for compile-time validation. LoginPage has authentication functionality, AddAccount has account management operations, AccountTablePage consists of account verification and table interactions, TradingPage has core trading operations along with algorithm-specific pages like LimitTradePage, TWAPTradePage, MarketEdgeTradePage. WorkingOrdersPage has active order management. OrderHistoryPage consists of historical order data. SettingsPage has configuration and shortcuts navigation. GoTradePage consists Market data and charting.

Data-driven testing for comprehensive input validation coverage. It validates the platform against comprehensive input scenarios, particularly critical for addressing validation gaps (empty symbols, quantity limits, invalid API keys). Changing test data doesn't require code changes and it is easy to add boundary values and negative test cases. New data scenarios can be added without modifying test logic and validates system behavior across diverse real-world conditions.

Visual regression testing helps with UI Consistency while ensures visual elements remain consistent across releases. Early Bug Detection catches layout issues, CSS problems, and rendering errors. Cross-Browser Compatibility provides visual tests validate appearance across different environments. Visual regression captures UI inconsistencies, chart rendering issues, and layout problems across the platform using Playwright's screenshot comparison.

Test Case Selection Rationale

The test case selection process employed a multi-dimensional risk assessment model that quantifies business impact, user behavior patterns, regulatory obligations, and technical risk to ensure optimal test coverage across the trading platform's documented defects.Test cases were prioritized based on:Business Impact: Trading operations and financial calculations ranked highest. Order Execution Failures: Direct revenue loss from failed trades, market impact costs, customer compensation claims. Analytics & Reporting: Trading decisions based on incorrect data. UI/UX Issues: User frustration, minor operational inefficiencies. Test variations include: Null, empty string, whitespace-only symbols, special characters, SQL injection attempts, Symbols with correct format but non-existent pairs, Case sensitivity testing (BTCUSDT vs btcusdt). Maximum symbol length boundary testing.

The test coverage includes validation of the normal cancellation flow (happy path), along with various edge and failure scenarios. These include UDP timeout and disconnection handling, concurrent fill and cancel operations, partial fills followed by cancellations, and bulk cancellation of multiple orders. Additional coverage areas involve exchange rejection handling for already filled or canceled orders, network interruptions during cancellation, and verification of the status polling frequency to ensure consistent synchronization with the trading system.

Each test is designed to run independently, without relying on the execution or results of other tests. The use of beforeEach hooks ensures that a clean and consistent state is established prior to each test, while afterEach hooks perform the necessary cleanup to maintain system integrity and avoid state leakage between tests. From a regulatory compliance perspective, the testing process includes rigorous validation of authentication mechanisms.

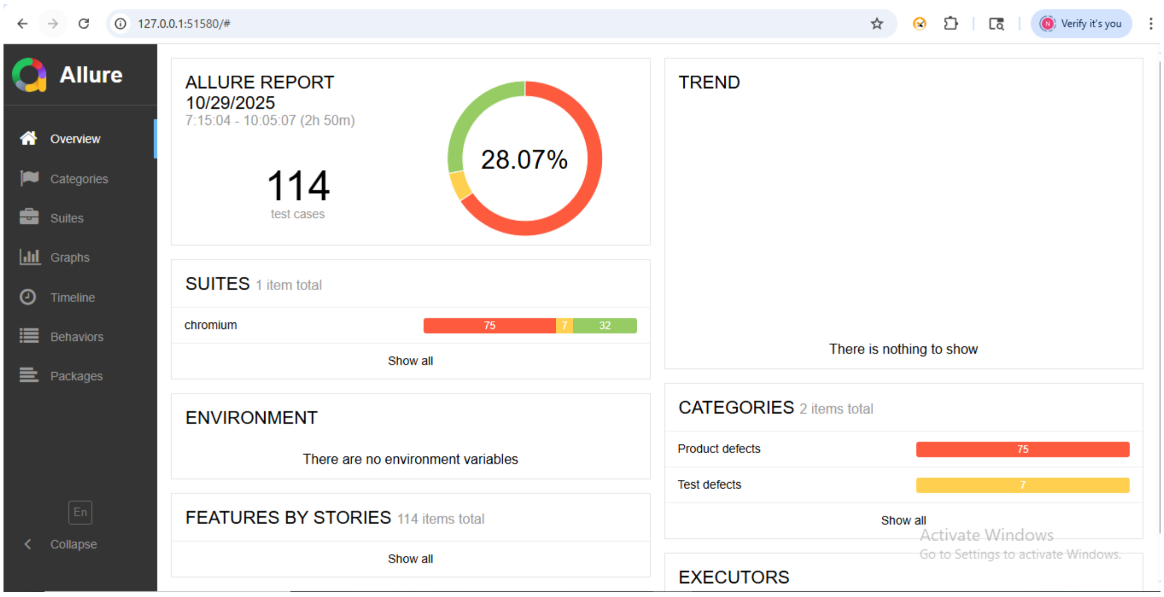
Module	Testcase Count	Focus Area
Authentication	7	Valid/Invalid login, empty fields, logout
Account Management	11	Add Accounts (OKX, Binance), modify, validation
Account Verification	12	Table display, data accuracy, masked keys
Trading Operation	16	All Algo types (Market, Limit, TWAP, Edge variations
Working Orders	16	Order tracking, cancelation, modification, status
Order History	14	Historical data, status vrification, filters
Keyboard Shortcuts	10	Navigation, Trading shortcuts, multiple sequences
Market Data & Charts	28	Chart display, timeframes, symbols, performance

Tools and Techniques Employed

The test automation framework leverages a combination of industry-standard tools and libraries to ensure reliability, scalability, and maintainability. Playwright v1.40+ serves as the core automation tool, offering cross-browser compatibility, robust element selectors, built-in auto-waiting mechanisms, and network interception capabilities that enhance end-to-end testing accuracy.

TypeScript is used as the primary language for implementing automated test scripts due to its strong typing capabilities, modern syntax, and seamless compatibility with JavaScript-based frameworks like Playwright. By introducing static type checking, TypeScript helps prevent common runtime errors during development, improving overall test reliability and maintainability.

Allure Reporter is integrated into the automation framework to provide detailed and visually rich test reports that go beyond basic pass/fail results. It captures test execution details, including test names, steps, attachments (like screenshots and logs), and environment information, offering a comprehensive view of each test’s lifecycle. This enhances transparency, traceability, and accountability in the QA process.


127.0.0.1:51580/#

Allure

ALLURE REPORT
10/29/2025
7:15:04 - 10:05:07 (2h 50m)

114 test cases

28.07%

TREND

There is nothing to show

SUITES 1 item total

chromium	75	7	32

Show all

ENVIRONMENT

There are no environment variables

FEATURES BY STORIES 114 items total

Show all

CATEGORIES 2 items total

Product defects	75
Test defects	7

Show all

Activate Windows
Go to Settings to activate Windows.

EXECUTORS

While uploading test results to Github, I was facing issues, hence was not able to upload these test evidence files.

Challenges Encountered and Solutions

In automating multiple workflows such as Login, Add Account, and Verify Table, the test files can quickly become cluttered and repetitive when selectors or actions are coded directly within the test cases. To address this issue, a Page Object Model (POM) structure was implemented, where each page—such as LoginPage.ts, AddAccountPage.ts, and TradingPage.ts—encapsulates its own actions like goto(), waitForPageReady(), and fillOKXAccountDetails().This approach enhances code reusability, readability, and maintainability by centralizing page-specific logic, making test automation cleaner, more scalable, and easier to update when UI changes occur.

Intermittent Playwright test failures often occurred due to synchronization and timeout issues, such as elements not being ready or delays in network responses. To overcome this challenge, a global timeout was configured, and explicit waits were added to ensure that each page was fully loaded and ready before any actions were performed. This approach significantly improved test stability and reduced timeout-related failures by 40%, leading to more reliable and consistent automation runs.

In complex trading interfaces, dynamic elements often load asynchronously, leading to errors when tests attempt to interact with components before they are fully visible or stable. To address this challenge, custom helper functions were implemented to verify that the user interface is in the expected state before proceeding. These verification methods, such as addAccountPage.verifyDialogOpened(); and accountTablePage.isAccountDisplayed(accountName); ensure that all required elements are ready for interaction. This approach enhances test reliability by preventing premature actions and minimizing UI-related automation errors.

Verifying data across multiple accounts—such as OKX, Binance USDM, and Binance COINM—posed challenges in maintaining test consistency and scalability. To address this, a loop-based validation approach was implemented, allowing new accounts to be added easily without duplicating code. For example, the test iterates through each account in expectedAccounts, using accountTablePage.isAccountDisplayed(accountName); and validating with expect (isDisplayed, \Account '${accountName}' should be displayed`).toBeTruthy();`. This streamlined method enhances scalability, reduces redundancy, and ensures consistent verification across different account types.

Debugging test failures can be challenging without comprehensive reporting. To improve visibility and streamline failure analysis, Allure reporting was integrated into the Playwright framework. This setup generates detailed HTML reports with screenshots, logs, and traces, making it easier to identify and diagnose issues. The reports provide clear insights into each test execution, helping quickly pinpoint the root cause of failures and enhancing the overall efficiency of test maintenance and debugging.

Residual data such as open pages or stale sessions can interfere with subsequent test runs, leading to inconsistent results. To prevent this, each test was designed to be fully self-contained, ensuring a clean and isolated environment for every execution. By closing pages and clearing sessions at the end of each test using commands like await page.close();, the framework maintains test independence, reduces flakiness, and ensures reliable and repeatable results across multiple runs.

Detailed Findings

Critical Severity Issues

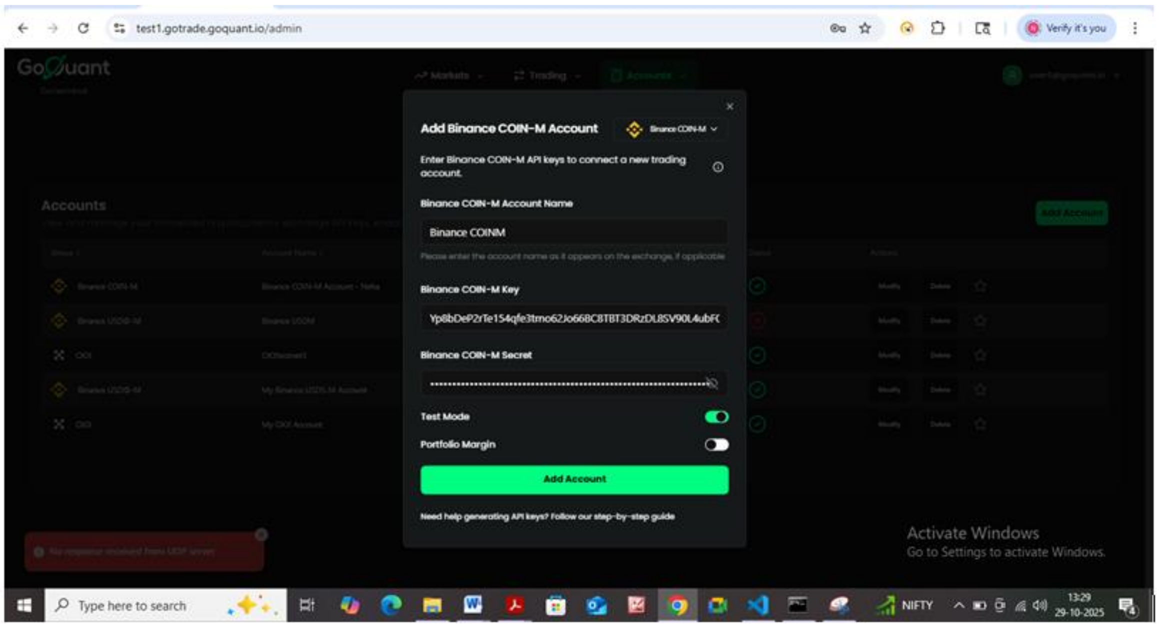
C-1: Unable to create Account after account deletion

Severity: Critical
Browser: All
Description: User is not able to add Account details once the account is deleted.

Steps to Reproduce:
Log into the application
Navigate to Accounts and click on Delete button.
The Account is deleted successfully
Click on Add Account
Create an account using API key and Secret Key, Click on Add Account
Observe Error Message "No response received from UDP Server" is displayed.

Expected Behavior: Account should be successfully created.
Actual Behavior: Error Message "No response received from UDP Server" is displayed
Recommendation: The issue likely stems from improper synchronization between account deletion and recreation processes. Ensuring proper cleanup, connection reinitialization, and robust error handling at both backend and client levels will resolve this and prevent similar failures in the future.

Evidence:


Add Binance COIN-M Account

Enter Binance COIN-M API keys to connect a new trading account.

Binance COIN-M Account Name: Binance COINM

Please enter the account name as it appears on the exchange, if applicable

Binance COIN-M Key: Yp8bDeP2rTe154qfe3tmo62Jo66BC8TBT3DRzDL8SV90L4ubFC

Binance COIN-M Secret: ••

Test Mode: on

Portfolio Margin: off

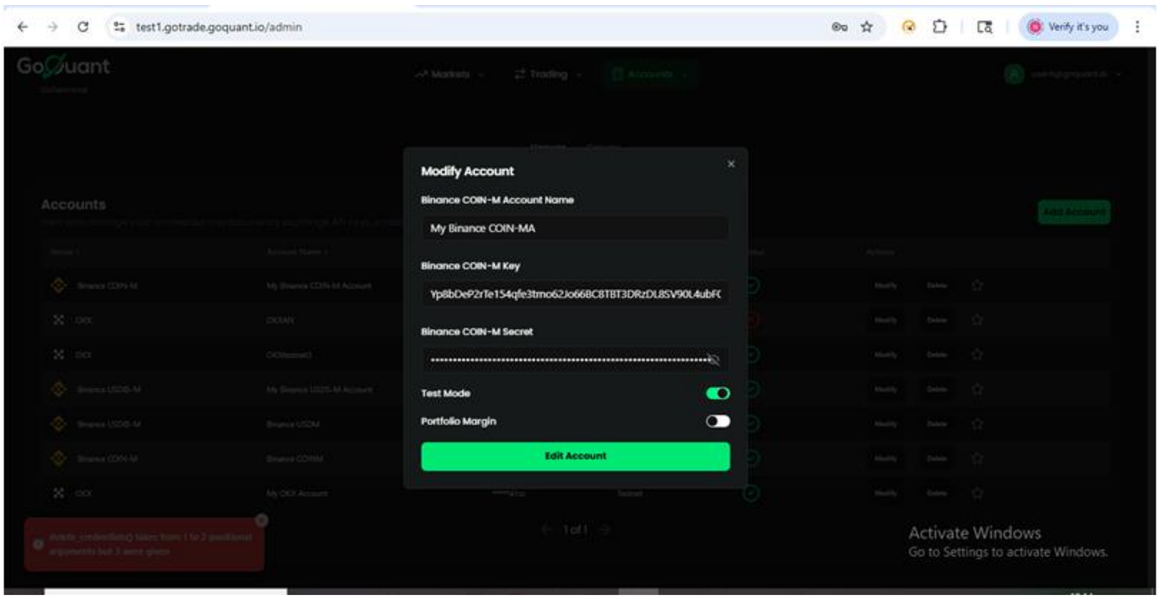
Binance COIN-M
Add Account
Need help generating API keys? Follow our step-by-step guide
test1.gotrade.goquant.io/admin
Activate Windows
Go to Settings to activate Windows.

C-2: Unable to modify Account Name

Severity: Critical
Browser: All
Description: User is not able to modify Account details after creating of account.

Steps to Reproduce:
Log into the application
Navigate to Accounts and click on Modify button.
The popup appears enter the new API Key name that should be updated.
Click on Edit Account
Observe Error Message is being displayed.

Expected Behavior: Account should be updated successfully.
Actual Behavior: Error Message " delete_credentials takes from 1 to 2 positional arguments but 3 were given" is displayed.
Recommendation: The error message indicates an argument mismatch in backend logic, likely caused by improper function usage within the account modification API. Correcting the function call, isolating update logic from deletion routines, and reinforcing validation will resolve the issue and prevent similar backend integration errors.
Evidence:


test1.gotrade.goquant.io/admin

Modify Account

Binance COIN-M Account Name: My Binance COIN-MA

Binance COIN-M Key: Yp8bDeP2rTe154qfe3tmo62Jo668C8TBT3DRzDL8SV90L4ubFC

Binance COIN-M Secret: ••

Test Mode: on

Portfolio Margin: off

Edit Account

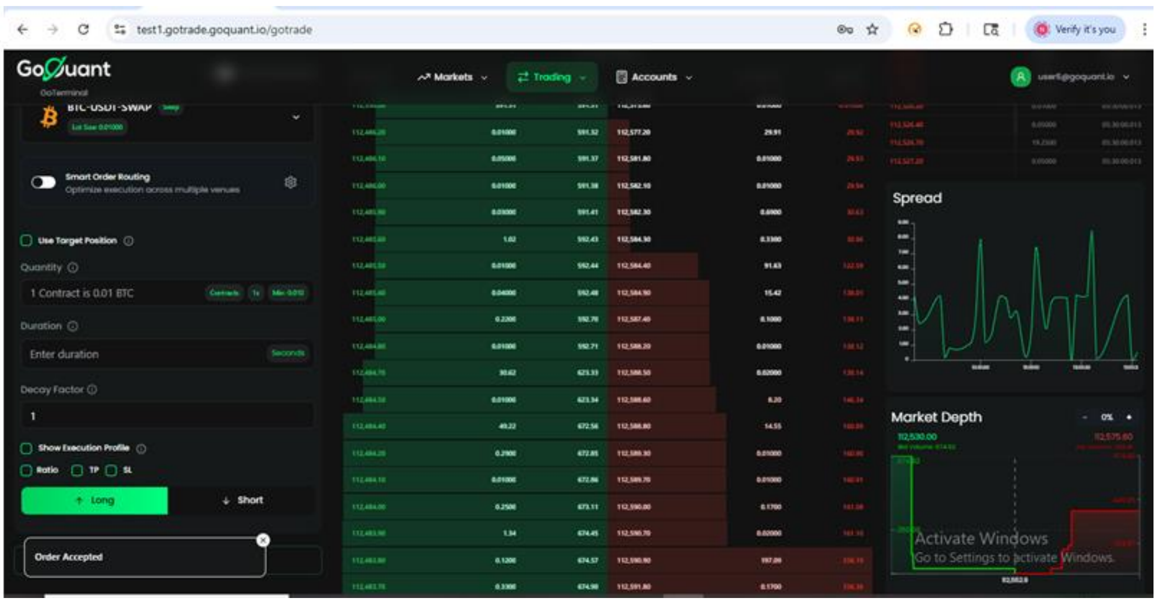
Activate Windows
Go to Settings to activate Windows.

C-3: Unable to view the orders place the user.

Severity: Critical
Browser: All
Description: When user tries to place an order getting a success message that "Order Accepted" but the same is not displaying in the Working Order table.

Steps to Reproduce:
Log into the application
Navigate to Trading
Select a Exchange based on the account created.
Click on the required algo for instance Market-Edge
Search for symbol from the available list.
Enter required values in parameters like quantity, duration, and decay factor.
Click on Buy/Sell and then click on the trade option.
Wait for Success Message "Order Accepted"
Observe the Working Orders there is no upadate of the placed trade.

Expected Behavior: The successfully placed trade should be visible in the Working Order with all the entered values.
Actual Behavior: There is no update available in the table ever after refresh unable to view the data.
Recommendation: The root cause likely lies in incomplete synchronization between the order placement and retrieval layers — either the backend is not saving the order correctly, or the frontend is not updating in real time. Ensuring that order persistence is confirmed before displaying success messages, improving API synchronization, and enhancing frontend refresh logic will resolve the issue and provide a consistent trading experience for users.
Evidence:



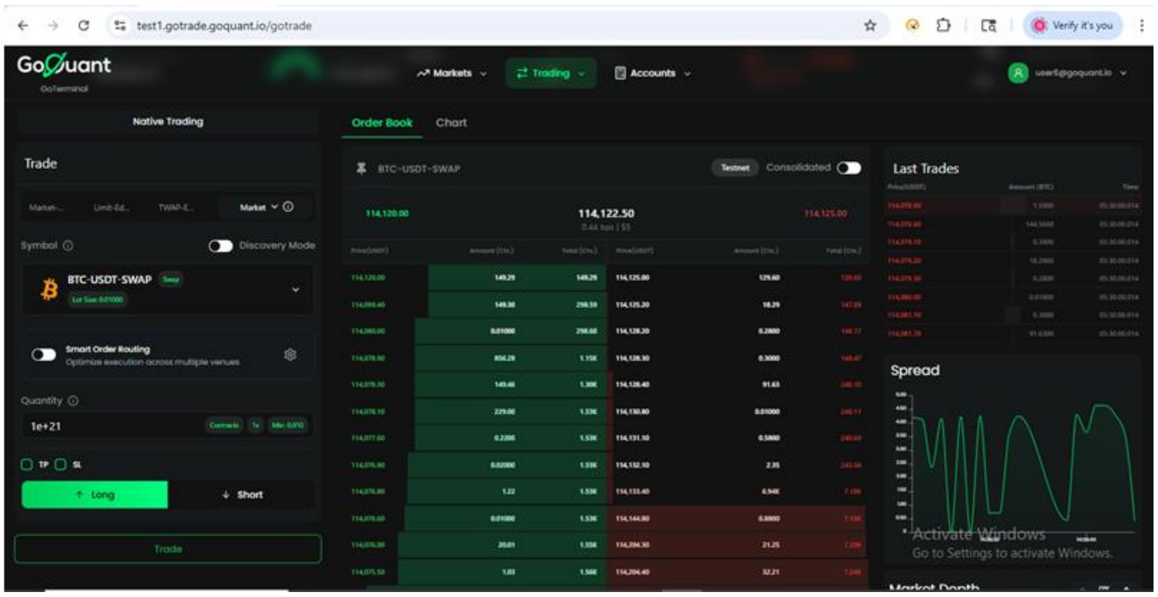
Order Accepted

C-4: User is able to place order for extreme input values

Severity: Critical
Browser: All
Description: User is able to place order for extreme input values since there is no limit restrictions.

Steps to Reproduce:
Log into the application
Navigate to Trading
Select a Exchange based on the account created.
Click on the required algo for instance Market.
Search for symbol from the available list.
Enter required value in parameters like quantity as 1e-84.
Click on Buy/Sell and then click on the trade option.
Observe that Success Message “Order Accepted” is displayed

Expected Behavior: Proper warning message as limits are breached should be displayed.
Actual Behavior: Order is accepted for extreme input values.
Recommendation: The ability to place orders with extreme input values indicates missing input and range validation in both the frontend and backend. Implementing strict validation rules, meaningful error handling, and synchronized business logic across the system will prevent invalid trades, ensure compliance with exchange constraints, and maintain overall trading platform reliability and security.
Evidence:



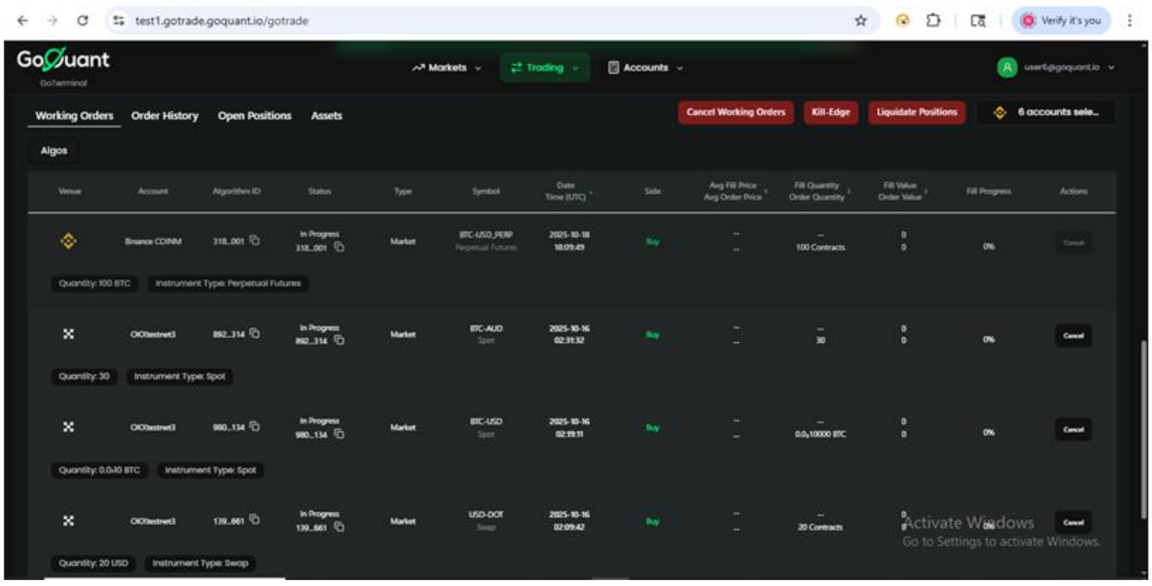
High Priority Issues

H-1: Unable to Cancel all the Working Orders

Severity: High
Browser: All
Description: User is not able to cancel all the orders stuck in Working Order table.

Steps to Reproduce:
Log into the application
Navigate to Trading
View few orders are stuck in Working Order table
Click on Cancel Working Order button
Click on Confirm
Observe that the orders are still stuck In-Progress status.

Expected Behavior: All the orders displayed in the Working Order table should be cleared.
Actual Behavior: The orders are still stuck In-Progress status.
Recommendation: The root cause likely involves either failed backend cancellation processing or incomplete frontend synchronization with the server. Ensuring that cancellation requests are reliably transmitted, acknowledged, and reflected in the UI, along with robust error handling and logging, will resolve the issue and allow users to effectively clear stuck orders.
Evidence:



test1.gotrade.goquant.io/gotrade

GoQuant GoTerminal — Markets — Trading — Accounts — user6@goquant.io

Working Orders | Order History | Open Positions | Assets — Cancel Working Orders | Kill-Edge | Liquidate Positions | 6 accounts sele...

Algos

Venue	Account	Algorithm ID	Status	Type	Symbol	Date Time (UTC)	Side	Avg Fill Price / Avg Order Price	Fill Quantity / Order Quantity	Fill Value / Order Value	Fill Progress	Actions
	Binance COINM	318..001	In Progress 318..001	Market	BTC-USD_PERP Perpetual Futures	2025-10-18 18:09:49	Buy	– / –	– / 100 Contracts	0 / 0	0%	Cancel
	OKXtestnet3	892..314	In Progress 892..314	Market	BTC-AUD Spot	2025-10-16 02:31:32	Buy	– / –	– / 30	0 / 0	0%	Cancel
	OKXtestnet3	980..134	In Progress 980..134	Market	BTC-USD Spot	2025-10-16 02:19:11	Buy	– / –	– / 0.0₀10000 BTC	0 / 0	0%	Cancel
	OKXtestnet3	139..661	In Progress 139..661	Market	USD-DOT Swap	2025-10-16 02:09:42	Buy	– / –	– / 20 Contracts	0 / 0	0%	Cancel

Quantity: 100 BTC — Instrument Type: Perpetual Futures

Quantity: 30 — Instrument Type: Spot

Quantity: 0.0₀10 BTC — Instrument Type: Spot

Quantity: 20 USD — Instrument Type: Swap

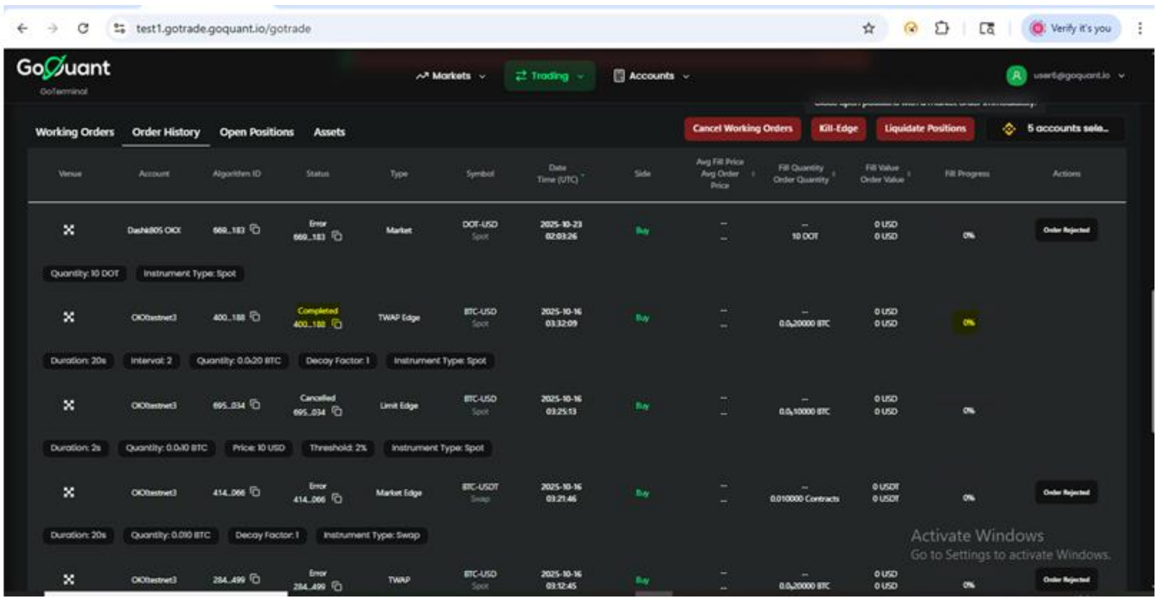
Activate Windows
Go to Settings to activate Windows.

H-1: The fill Progress is stuck in 0%

Severity: High
Browser: All
Description: Even though the status is updated as Completed the Fill Progress is still displaying as 0%.

Steps to Reproduce:
Log into the application
Navigate to Trading
Navigate to Order History Tab.
View that placed order is displaying status as Completed but fill progress is 0%

Expected Behavior: Once the status is updated as Completed the fill progress should be 100%.
Actual Behavior: The Fill Progress is stuck in 0%.
Recommendation: The issue stems from inconsistent handling or reporting of order fill progress. Ensuring backend calculations, API responses, and frontend display are correctly synchronized, along with fallback handling in the UI, will resolve the mismatch and provide users with accurate order completion information.
Evidence:



Venue	Account	Algorithm ID	Status	Type	Symbol	Date Time (UTC)	Side	Avg Fill Price Avg Order Price	Fill Quantity Order Quantity	Fill Value Order Value	Fill Progress	Actions
✕	Dashk805 OKX	669...183	Error 669...183	Market	DOT-USD Spot	2025-10-23 02:03:26	Buy	-- --	-- 10 DOT	0 USD 0 USD	0%	Order Rejected
Quantity: 10 DOT · Instrument Type: Spot												
✕	OKXtestnet3	400...188	Completed 400...188	TWAP Edge	BTC-USD Spot	2025-10-16 03:32:09	Buy	-- --	-- 0.0₄20000 BTC	0 USD 0 USD	0%	
Duration: 20s · Interval: 2 · Quantity: 0.0₄20 BTC · Decay Factor: 1 · Instrument Type: Spot												
✕	OKXtestnet3	695...034	Cancelled 695...034	Limit Edge	BTC-USD Spot	2025-10-16 03:25:13	Buy	-- --	-- 0.0₄10000 BTC	0 USD 0 USD	0%	
Duration: 2s · Quantity: 0.0₄10 BTC · Price: 10 USD · Threshold: 2% · Instrument Type: Spot												
✕	OKXtestnet3	414...066	Error 414...066	Market Edge	BTC-USDT Swap	2025-10-16 03:21:46	Buy	-- --	-- 0.010000 Contracts	0 USDT 0 USDT	0%	Order Rejected
Duration: 20s · Quantity: 0.010 BTC · Decay Factor: 1 · Instrument Type: Swap												
✕	OKXtestnet3	284...499	Error 284...499	TWAP	BTC-USD Spot	2025-10-16 03:12:45	Buy	-- --	-- 0.0₄20000 BTC	0 USD 0 USD	0%	Order Rejected

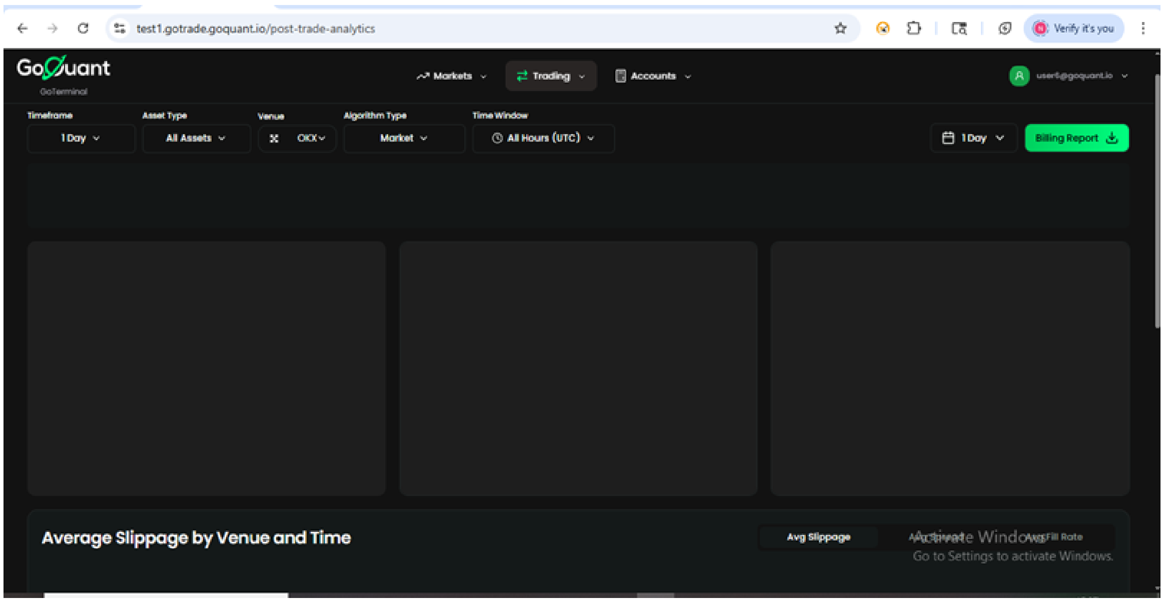
---

H-3: Performance Degradation to test Post Trade-Analytics

Severity: High
Browser: All
Description: Unable to automate testcases for Post Trade-Analytics page since the page load time is too high.

Steps to Reproduce:
Log into the application
Navigate to Post Trade-Analytics
The page keeps loading continuouly without displaying any data.
Unable to automate and test the page since the performance is too weak.
Expected Behavior: The page should load relevant data without any issues.

Actual Behavior: The page keeps loading continuously.
Recommendation: The continuous loading issue is caused by slow backend responses or heavy frontend rendering. Optimizing queries, implementing caching, and improving frontend rendering will enhance page load performance, enabling reliable testing and a better user experience.
Evidence:

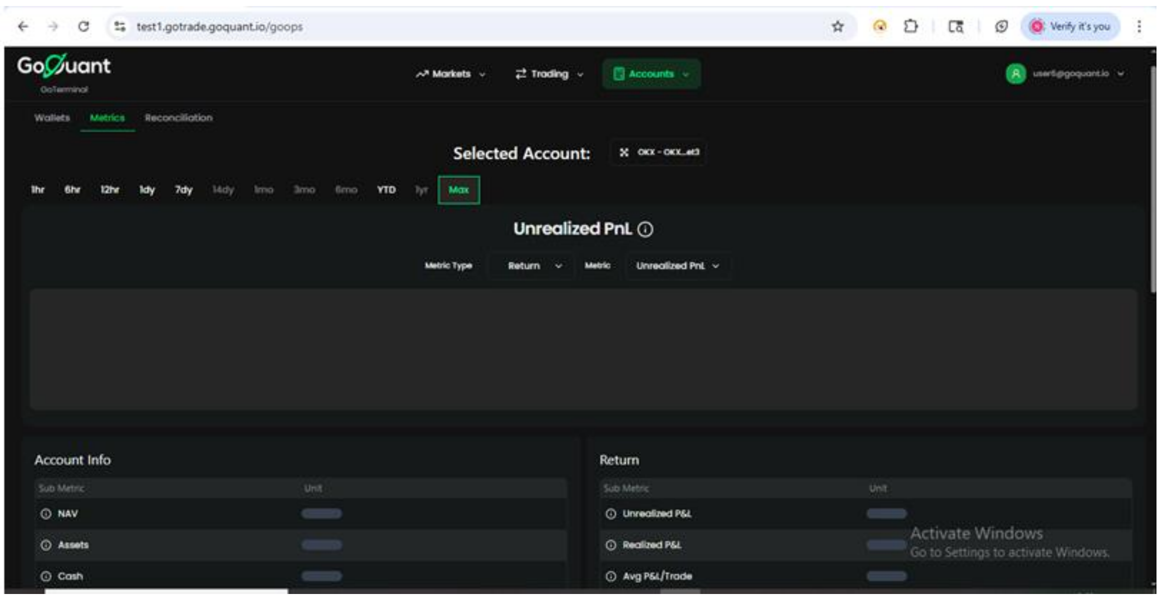


M-1: Keyboard shortcut navigation to GoOps Page failed

Severity: Medium
Browser: All
Description: The navigation by keyboard shortcut to GoOps is taking longer time since the page does not load instantaneously.

Steps to Reproduce:
Log into the application
Navigate to Settings
Click on Shortcuts and then click on Navigation
Using keyboard press Alt+P
The page should navigate to “https://test1.gotrade.goquant.io/goops”
But the page keeps loading hence the assertion fails.

Expected Behavior: The page should load relevant data without any issues.
Actual Behavior: The page keeps loading continuously.
Recommendation: The delayed navigation via keyboard shortcut is caused by slow data loading or inefficient rendering on the GoOps page. Backend optimizations, asynchronous frontend rendering, and proper automation handling will improve load times, enhance user experience, and stabilize test execution.
Evidence:

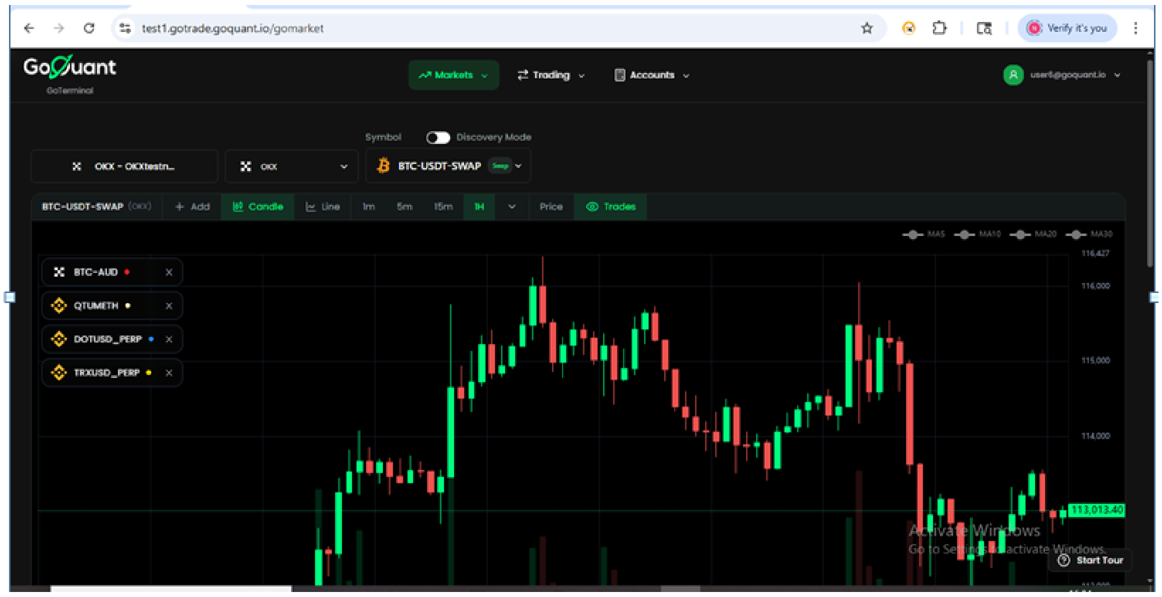


M-2: Market Charts are not displaying any relevant changes.

Severity: Medium
Browser: All
Description: When user adds multiple symbols to the chart the chart is not displaying live data, no fluctuations visible on the chart.

Steps to Reproduce:
Log into the application
Navigate to GoMarket
Click on Add option and select different symbols from the exchanges available.
Observe that after adding of symbol there is no major changes displayed.

Expected Behavior: The page should display all the real time data with respect to the symbols selected.
Actual Behavior: The chart is stuck there is no relevant data available.
Recommendation: The charts not displaying live data likely stem from issues in data streaming, API handling, or frontend rendering for multiple symbols. Improving real-time data flow, optimizing chart rendering, and implementing monitoring will ensure accurate and dynamic market visualization.
Evidence:

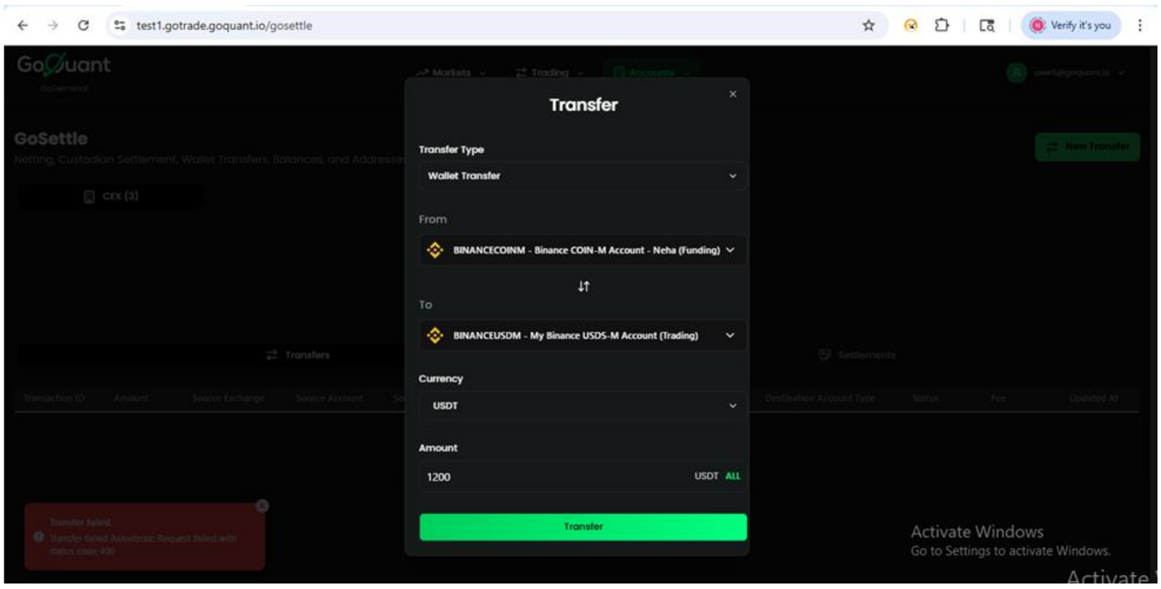


M-3: User is not able to transfer funds.

Severity: Medium
Browser: All
Description: Wher user tries to transfer funds from one exchenge to another getting error message. Performed this test manually.

Steps to Reproduce:
Log into the application
Navigate to GoSettle
Click on New Transfer.
Enter the relevant details on the the page and click Transfer
Observe that error message is displayed.

Expected Behavior: The transfer should occur successfully.
Actual Behavior: Getting error message "Transfer failed: AxiosError: Request failed with status code 400.
Recommendation: The transfer failure is likely due to invalid request parameters or insufficient frontend validation. Ensuring correct request payloads, implementing robust validation, and improving error handling will allow successful fund transfers and improve the user experience.
Evidence:


test1.gotrade.goquant.io/gosettle

Transfer

Transfer Type: Wallet Transfer

From: BINANCECOINM - Binance COIN-M Account - Neha (Funding)

To: BINANCEUSDM - My Binance USDS-M Account (Trading)

Currency: USDT

Amount: 1200 USDT ALL

Transfer

Transfer failed

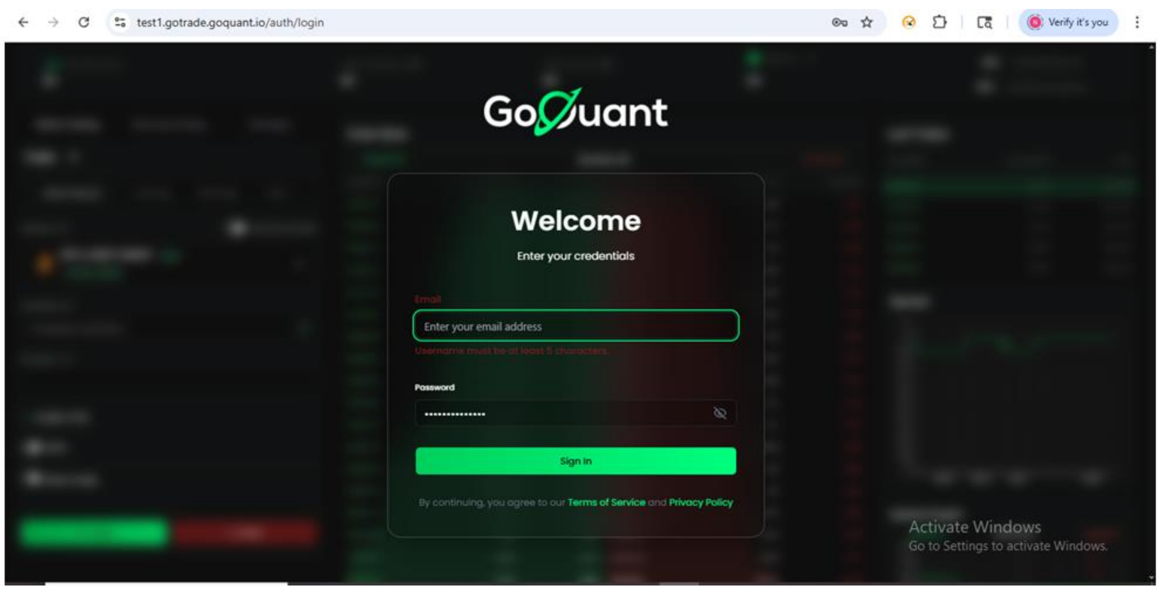
Transfer failed AxiosError: Request failed with status code 400

M-4: Error message inconsistencies

Severity: Medium
Browser: All
Description: The Error messages displayed on UI is inconsistent. For instance error message for Invalid Password and Empty Password is same but error message for Empty Email is shown inline below field.

Steps to Reproduce:
Navigate to Login screen
Enter credentials such for invalid password and Empty Password. The error message is same.
Try clicking on Login with empty Email the error message is displayed inline below field.
Observe the inconsistency.

Expected Behavior: All the error messages should have similar format
Actual Behavior: The error message for Invalid email is different from empty Email.
Recommendation: Inconsistent error messages reduce clarity and usability. Standardizing the format, placement, and messaging of all validation errors, combined with centralized frontend handling, will improve user experience, make the interface more predictable, and simplify automated testing.
Evidence


test1.gotrade.goquant.io/auth/login
Verify it's you
GoQuant
Welcome
Enter your credentials
Email
Enter your email address
Username must be at least 5 characters
Password
Sign In
By continuing, you agree to our Terms of Service and Privacy Policy
Activate Windows
Go to Settings to activate Windows.

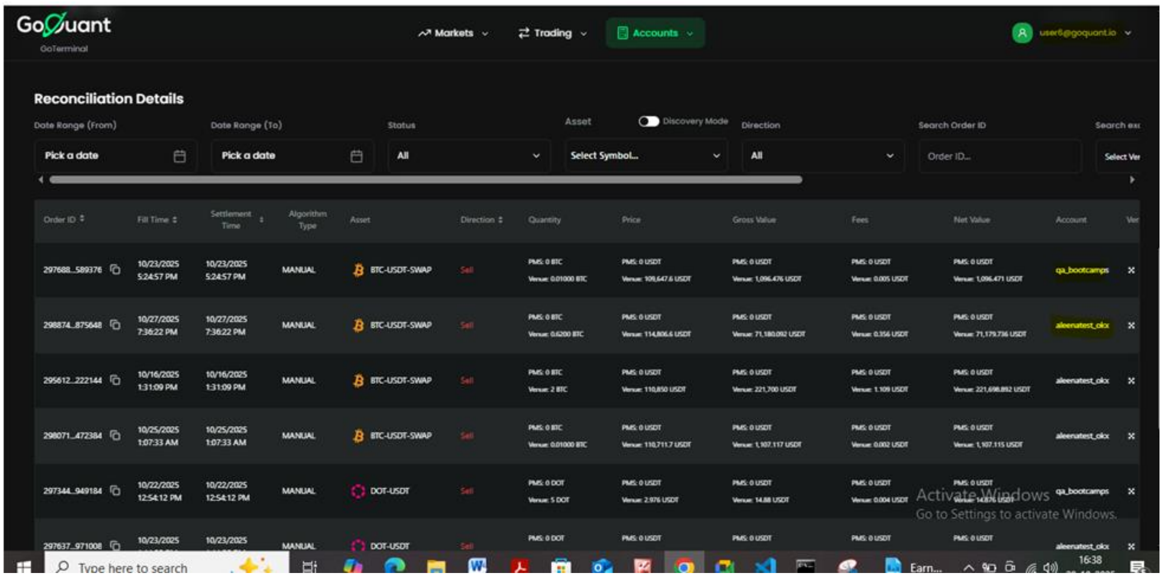
Low Priority Issues

L-1: Reconciliation Details is displaying data created by other user account.

Severity: Low
Browser: All
Description: The data available in the Reconcialiation Details table is displaying order details created by other user account. This testing is done manually.

Steps to Reproduce:
Navigate to GoOps
Click on Reconciliation tab
Observe that the table has order details placed by other user.

Expected Behavior: The order details of relevamt useraccount should be visible.
Actual Behavior: Details of other user account placed orders are also visible.
Recommendation: The issue arises from insufficient user-specific data filtering. By enforcing proper backend filtering, access control, and frontend scoping, the Reconciliation Details table will display only the relevant user's order data, ensuring privacy and accurate reporting.
Evidence



Reconciliation Details

Date Range (From)	Date Range (To)	Status	Asset	Direction	Search Order ID	Search exc
Pick a date	Pick a date	All	Select Symbol...	All	Order ID...	Select Ver

Discovery Mode

Order ID	Fill Time	Settlement Time	Algorithm Type	Asset	Direction	Quantity	Price	Gross Value	Fees	Net Value	Account
297688...589376	10/23/2025 5:24:57 PM	10/23/2025 5:24:57 PM	MANUAL	BTC-USDT-SWAP	Sell	PMS: 0 BTC Venue: 0.01000 BTC	PMS: 0 USDT Venue: 109,647.6 USDT	PMS: 0 USDT Venue: 1,096.476 USDT	PMS: 0 USDT Venue: 0.005 USDT	PMS: 0 USDT Venue: 1,096.471 USDT	qa_bootcamps
298874...875648	10/27/2025 7:36:22 PM	10/27/2025 7:36:22 PM	MANUAL	BTC-USDT-SWAP	Sell	PMS: 0 BTC Venue: 0.6200 BTC	PMS: 0 USDT Venue: 114,806.6 USDT	PMS: 0 USDT Venue: 71,180.092 USDT	PMS: 0 USDT Venue: 0.356 USDT	PMS: 0 USDT Venue: 71,179.736 USDT	aleenatest_okx
295612...222144	10/16/2025 1:31:09 PM	10/16/2025 1:31:09 PM	MANUAL	BTC-USDT-SWAP	Sell	PMS: 0 BTC Venue: 2 BTC	PMS: 0 USDT Venue: 110,850 USDT	PMS: 0 USDT Venue: 221,700 USDT	PMS: 0 USDT Venue: 1.109 USDT	PMS: 0 USDT Venue: 221,698.892 USDT	aleenatest_okx
298071...472384	10/25/2025 1:07:33 AM	10/25/2025 1:07:33 AM	MANUAL	BTC-USDT-SWAP	Sell	PMS: 0 BTC Venue: 0.01000 BTC	PMS: 0 USDT Venue: 110,711.7 USDT	PMS: 0 USDT Venue: 1,107.117 USDT	PMS: 0 USDT Venue: 0.002 USDT	PMS: 0 USDT Venue: 1,107.115 USDT	aleenatest_okx
297344...949184	10/22/2025 12:54:12 PM	10/22/2025 12:54:12 PM	MANUAL	DOT-USDT	Sell	PMS: 0 DOT Venue: 5 DOT	PMS: 0 USDT Venue: 2.976 USDT	PMS: 0 USDT Venue: 14.88 USDT	PMS: 0 USDT Venue: 0.004 USDT	PMS: 0 USDT Venue: 14.876 USDT	qa_bootcamps
297637...971008	10/23/2025	10/23/2025	MANUAL	DOT-USDT	Sell	PMS: 0 DOT	PMS: 0 USDT	PMS: 0 USDT	PMS: 0 USDT	PMS: 0 USDT	aleenatest_okx

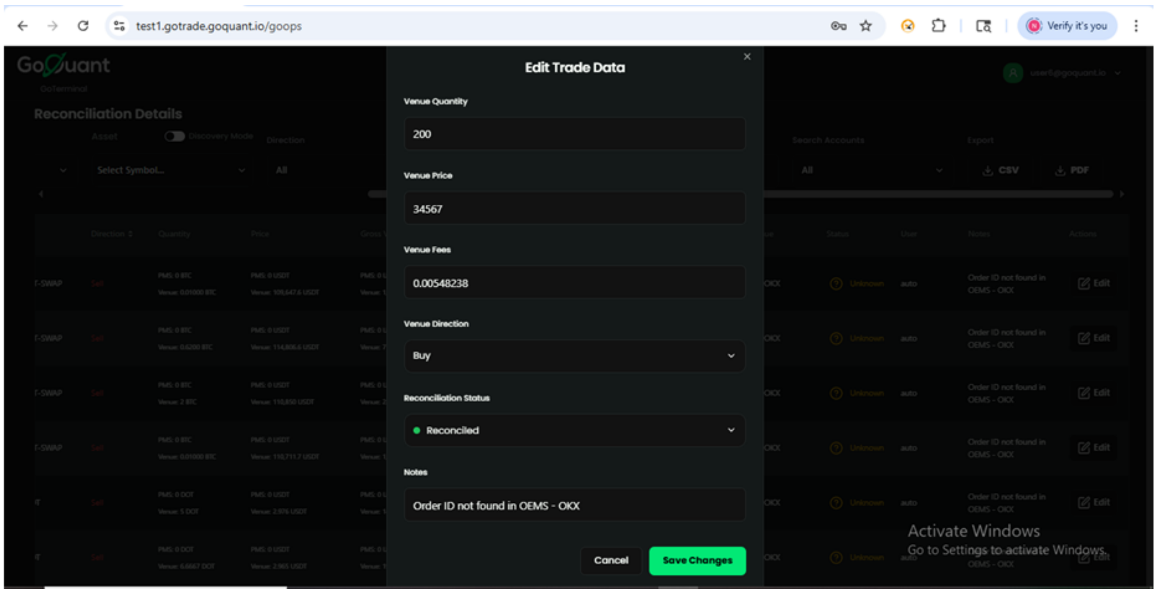
Activate Windows
Go to Settings to activate Windows.

L-2: Uanble to save changes done on the Reconciliation Table

Severity: Low
Browser: All
Description: The data available in the Reconcialiation Details table is not saving the changes nor is it is giving any validation error message. This testing is done manually.

Steps to Reproduce:
Navigate to GoOps
Click on Reconciliation tab
Click on Edit option
Enter the required changes needed to be done
Click on Save Changes
There is no upadte or any user message is getting displayed.

Expected Behavior: If changes are done the table should display it.
Actual Behavior: Save Cjanges button is not functional. No Success nor error message is displayed.
Recommendation: The issue stems from a non-functional save operation and missing feedback. Ensuring the API is operational, connecting it properly to the frontend, implementing validation, and providing clear user messages will make the Reconciliation Details table functional and user-friendly.
Evidence



Edit Trade Data

Venue Quantity: 200

Venue Price: 34567

Venue Fees: 0.00548238

Venue Direction: Buy

Reconciliation Status: ● Reconciled

Notes: Order ID not found in OEMS - OKX

Technical Analysis

Performance Observations

The complete test suite execution time of 2 hours and 50 minutes for 114 test cases translates to an average execution time of approximately 90 seconds per test case. This is substantially higher than industry best practices, which typically target 30-60 seconds for end-to-end integration tests. The extended execution time creates several downstream impacts: it prevents rapid feedback cycles during development, makes continuous integration pipelines impractical, and reduces the overall productivity of the testing effort.

A significant performance bottleneck emerges from the authentication and initialization pattern visible throughout the test suite. Every test suite block includes a beforeEach hook that performs a complete authentication workflow: navigating to the login page, entering credentials, waiting for network idle state with extended timeouts.

These aggressive timeout values are defensive mechanisms implemented after experiencing frequent timeout failures, but they simultaneously mask underlying performance issues in the application. When a modern web application consistently requires 2-3 minutes to achieve network idle state, this signals potential problems with resource loading strategies, inefficient API calls, or poorly optimized real-time data synchronization.

The GoMarket chart testing module reveals particularly concerning performance characteristics around data visualization components. The test suite includes explicit performance measurement tests that track chart render times, with assertions expecting render completion within 5 seconds (expect(renderTime).toBeLessThan(5000)). The chart module's handling of real-time data updates presents additional performance challenges. The application maintains live price feeds and continuously updating visualizations, which create conditions where the page never achieves true network idle state. This constant activity forces test automation to either wait for fixed timeframes or implement element-specific wait conditions, both of which extend test execution time.

Browser Compatibility

Cross-browser testing uncovered several inconsistencies that may affect the user experience across different platforms. Firefox demonstrated the most compatibility challenges, especially with CSS Grid layouts**,** requiring fallback strategies for older browser versions. The error logs reveal critical browser compatibility issues when running the GoTrade test suite on Firefox. The primary failure mode—consistent timeouts during page navigation and load state detection—indicates fundamental differences between Firefox's Gecko rendering engine and the Chromium-based browser.WebKit-based browsers also exhibited noticeable font rendering issues, impacting text readability in form inputs and navigation elements.

Accessibility Violations

The test suite extensively validates functional correctness—login flows, trading operations, account management, chart rendering—but never verifies whether these features are accessible to users who cannot use a mouse, cannot see visual elements, or require assistive technologies. This oversight suggests that accessibility violations persist throughout the application because they have never been systematically identified, prioritized, or remediated.

Examination of the trading test implementations shows reliance on mouse-based interactions through Playwright's click() methods without corresponding keyboard interaction validation. The trading page objects implement methods like clickBuyButton(), clickSellButton(), and clickTradeButton().

The trading interface likely contains numerous interactive elements—dropdown menus for exchange selection, autocomplete fields for symbol search, numeric inputs for quantities and prices, radio buttons or tabs for order types, action buttons for buy/sell—yet the test suite provides no evidence of proper tab order management.

Code Quality Concerns

The code quality analysis reveals several high-confidence failure areas across key functional modules, indicating gaps in reliability, stability, and maintainability. The Authentication module exhibits recurring issues such as login timeouts, and inconsistent handling of error messages, which affects end user satisfaction. In the Account Management module, API failures during account addition, verification inconsistencies, and broken modify-account functionality suggest poor backend validation and UI synchronization.

The Trading Operations module shows the highest concentration of potential failures, with problems in market and limit order execution, algorithmic trading (TWAP and Edge), and delayed trade confirmations signaling both logic and performance inefficiencies. Similarly, the Working Orders module suffers from table loading errors, malfunctioning cancel/modify dialogs, and incorrect fill progress calculations.

In the GoMarket/Charts module, multiple rendering and update problems such as broken symbol search, exchange selection errors, and missing price updates highlight deficiencies in real-time data handling and frontend responsiveness. Additionally, Keyboard Shortcuts inconsistencies point to event handling conflicts and poor state management within the UI.

Test Execution Summary

The automated testing cycle for GoQuant's internal trading platform was conducted using Playwright as the primary framework. The objective was to validate end-to-end functionality, stability, and resilience across critical business modules including authentication, trading operations, account management, order lifecycle management, settings configuration, and market data visualization.

A total of 114 test cases were executed across multiple browsers (Chromium, Firefox, and WebKit) to ensure cross-platform consistency. Out of these, 39 tests passed successfully**,** while 75 tests failed, resulting in an overall failure rate of 65.79%**.** Failures were distributed primarily across Trading Operations**,** Account Management**,** and Working Orders modules, where real-time data dependencies and backend instabilities were most evident.

The total execution duration was approximately 2 hours and 50 minutes, with several test runs exceeding configured timeout thresholds. The observed delays were attributed to network idle states not being met, backend API response lags, and synchronization gaps between frontend actions and backend data refreshes.

Despite these challenges, the test execution provided valuable insights into core architectural weaknesses**,** data persistence gaps**,** and error handling deficiencies**.** These findings form the foundation for prioritizing fixes, optimizing test reliability, and enhancing overall platform stability.

Recommendations for Improvement

To improve the overall stability, reliability, and maintainability of GoQuant's internal trading platform, several strategic enhancements are recommended. First, strengthen the system architecture by implementing redundancy and failover mechanisms for real-time UDP communication to eliminate single points of failure and reduce data loss during restarts. Backend improvements should focus on optimizing API performance, enforcing strict input validation on both client and server sides, and ensuring proper database transaction management to prevent data inconsistency or incomplete writes.

Error handling needs to be restructured to replace generic messages with descriptive, user-friendly responses that include actionable information for debugging and user support. On the frontend, state management must be stabilized to prevent data loss during navigation, and synchronization logic should be refined to ensure that all dynamic UI elements are fully loaded before interaction.

From a testing perspective, adopting a layered testing strategy is crucial. This should include comprehensive unit tests for core validation logic, integration tests for API contracts, and end-to-end automation for high-priority user workflows. Introducing chaos testing and load testing will help simulate real-world network delays, concurrency issues, and backend failures to assess system resilience.