KONFIDENTIELL

Intern / Management

TESTKONZEPT & TESTFALLSPEZIFIKATION

MT5 Multi-Backtester

Unit-Test Implementierung & QA-Strategie

73	4	13.5 PT	>85%
Unit Tests	Testdomänen	Aufwand	Abdeckung

Dieses Dokument definiert das vollstaendige QA-Framework fuer den MT5 Multi-Backtester. Es dient als verbindliche Implementierungsspezifikation fuer 73 kritische Unit-Tests und als belastbare Grundlage fuer die Aufwandskalkulation und Ressourcenplanung.

Projekt:	**MT5 Multi-Backtester**
Version:	**1.1 – Finale Freigabeversion**
Datum:	**07. Mai 2026**
Autor:	**Thomas Nickel / IT Development & QA**
Standard:	**ISTQB Foundation Level (FL-2.2, FL-3.2)**
Klassifizierung:	**Intern / Vertraulich**

Inhaltsverzeichnis

1. Management Summary

Dieses Testkonzept definiert die vollständige Qualitätssicherungsstrategie für den **MT5 Multi-Backtester**, eine Java-basierte Desktop-Applikation zur vollautomatischen Steuerung, Parametrisierung und Auswertung von MetaTrader 5 Expert Advisor (EA) Backtests. Als geschäftskritische Trading-Infrastruktur unterliegt die Anwendung höchsten Anforderungen an Zuverlässigkeit, Datenkonsistenz und numerische Präzision.

Das Konzept basiert auf den Qualitätsstandards des **ISTQB Foundation Level** (FL-2.2, FL-3.2) und definiert verbindliche Testfälle, Erfolgskriterien sowie den Ressourcenbedarf zur Erreichung einer produktionsreifen Testabdeckung von **>85% Line Coverage**.

Strategische Zielsetzung: Durch die Implementierung dieser 73 Unit-Tests werden regressionsfreie Releases sichergestellt, die Fehlerentdeckungsrate in der frühen Entwicklungsphase signifikant erhöht (Faktor 10x günstiger als Produktionsfehler) und eine stabile CI/CD-Pipeline etabliert, die Pull Requests automatisch validiert.

Tabelle 1: Qualitäts-KPIs und Zielwerte

KPI	Zielwert	Messmethode	Status
Line Coverage (Engine + Report)	> 85 %	JaCoCo Report	Ziel
Branch Coverage (Kritischer Pfad)	> 75 %	JaCoCo Report	Ziel
Test-Ausführungszeit (gesamt)	< 15 Sek.	CI Pipeline Log	Ziel
Fehlgeschlagene Tests (Produktion)	0	GitHub Actions	Ziel
Kritische Bugs in Produktion (nach Release)	0	Bug-Tracker	Ziel

2. Projekthintergrund & Systemarchitektur

Der MT5 Multi-Backtester ist eine Java/Swing Desktop-Applikation mit 8 Kernmodulen, 30+ Java-Klassen und >15.000 Zeilen Code. Die Anwendung orchestriert vollautomatisch den MetaTrader 5 Terminal-Prozess per CLI und INI-Konfiguration und ersetzt hunderte manuelle Klicks durch einen vollständig automatisierten Pipeline-Prozess.

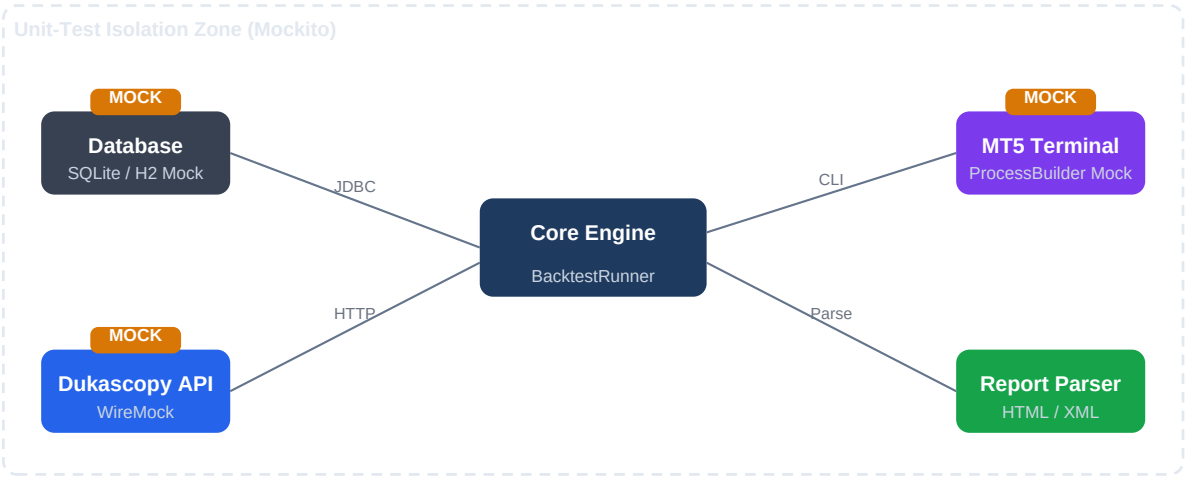

Abbildung 3: Testarchitektur & Mocking-Strategie

Tabelle 2: Systemmodule und Testprioritäten

Modul	Beschreibung	Test-Priorität
Single Backtester	Einzelläufe mit konfigurierbaren Parametern, EA-Set-File Editor	HOCH
Multi-Batch Runner	Sequentielle N×M×X Kombinationsläufe mit Fault Tolerance	HOCH
MT5 Optimizer	Genetischer & vollständiger Algorithmus-Optimizer mit Forward Testing	HOCH
Robustness Scanner	Parametersensitivitätsanalyse mit historischen Time-Shifts	HOCH
EA Config Mgmt.	Read/Write für MT5 UTF-16LE .set Dateien, DB-Snapshots	MITTEL
History & Persistence	SQLite-Persistenz, Master-Detail-Tree, Session-Browser	HOCH
Dukascopy Integration	HTTP Download, BI5 LZMA-Dekompression, CSV-Konvertierung	HOCH
Reporting Engine	MT5 HTML/XML Parser, HTML-Summaries, PDF Export	HOCH

3. Teststrategie & Methodik

3.1 Test-Framework & Werkzeuge

Tabelle 3: Test-Framework & Werkzeuge

Werkzeug	Version	Einsatzzweck	Begründung
JUnit 5 (Jupiter)	5.10+	Test-Runner & Annotations	De-facto Standard für moderne Java-Applikationen; unterstützt parametrisierte Tests, Lifecycle-Extensions und parallele Ausführung.
Mockito + Mockito-Inline	5.x	Mocking externer Abhängigkeiten	Erlaubt das Mocken von Klassen (nicht nur Interfaces) inkl. static methods. Kritisch für ProcessBuilder und DateiSystem-Mocks.
AssertJ	3.x	Fluent Assertions	Deutlich lesbarere Fehlermeldungen als JUnit-Standard. Ermöglicht domänenspezifische Assertions für Listen, Files und numerische Vergleiche.
JaCoCo	0.8.x	Code-Coverage-Analyse	Maven-Plugin-Integration, HTML/XML Reports, Branch & Line Coverage. Konfiguriert als Build-Breaker bei <85% auf kritischen Packages.
WireMock	3.x	HTTP-Mocking (Dukascopy)	Embedded HTTP-Server zur Simulation von Dukascopy API-Responses. Kein Netzwerkverkehr in Tests.
H2 Database	2.x	In-Memory-DB für Tests	SQLite-kompatibler In-Memory-Modus. Jeder Test erhält eine frische, isolierte DB-Instanz in unter 5ms.

3.2 Test-Prinzipien & Architektur-Grenzen

Die Mocking-Strategie folgt dem Prinzip der strikten Isolation: Jede externe Abhängigkeit wird durch eine kontrollierte Test-Double ersetzt. Die drei kritischen Grenzen sind:

- **MT5 Terminal (ProcessBuilder)**: Der Aufruf von `terminal64.exe` in BacktestRunner, OptimizationRunner und RobustnessRunner wird per Mockito gemockt. Simulierte MT5 HTML/XML-Reports werden aus dem Test-Ressourcenverzeichnis geladen.
- **SQLite Datenbank**: Der DatabaseManager wird gegen eine H2-In-Memory-Datenbank getestet. Jede @Test-Methode erhält via @BeforeEach eine frische, leere Datenbankinstanz.
- **Dukascopy HTTP API**: Der DukascopyDownloader wird gegen einen eingebetteten WireMock-Server getestet. Statische .bi5-Binärdaten aus dem Test-Ressourcenpfad simulieren echte API-Responses ohne Netzwerkabhängigkeit.

FIRST-Prinzip (ISTQB FL-3.2): Alle Tests entsprechen dem FIRST-Standard: **F**ast (Gesamtlaufzeit <15s), **I**ndependent (keine Abhängigkeiten zwischen Tests), **R**epeatable (deterministisch auf Windows & Linux CI), **S**elf-Validating (automatische Pass/Fail-Entscheidung), **T**imely (vor Produktionsmerge implementiert).

Namenskonvention: Alle Test-Methoden folgen dem Muster `methode_ausgangszustand_erwartetesErgebnis`, z.B. `generateSetFile_validParameters_createsFileSuccessfully`. Dies gewährleistet unmittelbare Lesbarkeit im CI-Log ohne Kommentare.

3.3 Testprozess-Flow

Abbildung 1: Testprozess-Flow nach ISTQB FL-2.2

3.4 Fehlerlebenszyklus

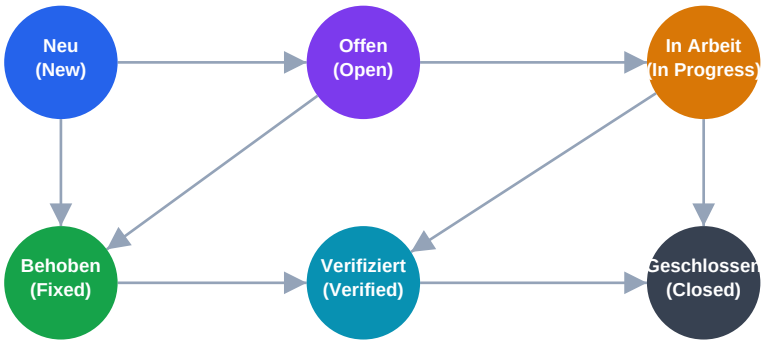

Abbildung 2: Fehlerlebenszyklus (Defect Lifecycle)

4. Testumgebungen & Infrastruktur

Tabelle 4: Testumgebungsmatrix

Umgebung	Typ	OS	JDK	Datenbank	MT5	Zweck
Local Dev	Entwickler-PC	Windows 11	OpenJDK 21	H2 In-Memory	MOCK	Lokale TDD-Entwicklung
CI Pipeline	GitHub Actions	Ubuntu 22.04 LTS	OpenJDK 21	H2 In-Memory	MOCK	Automatische PR-Validierung
Integration	Staging VM	Windows Server 2022	OpenJDK 21	SQLite (File)	Real MT5 Demo	Manuelle Integration-Tests
Production	Anwender-PC	Windows 10/11	Bundled JRE	SQLite (File)	Real MT5 Live	Produktionsbetrieb

5. Testfallspezifikation – 73 Unit-Tests

Die nachfolgenden Testfälle bilden die Mindestanforderung für eine produktionsreife Freigabe. Jeder Testfall ist vollständig nach der **Given/When/Then**-Struktur spezifiziert und enthält konkrete Testdaten sowie das präzise erwartete Ergebnis. Die Reihenfolge entspricht der empfohlenen Implementierungsreihenfolge.

5.1 Core Engine & Backtest Logic (TC-ENG-01 bis TC-ENG-25)

Diese 25 Tests verifizieren die kritischen Kernfunktionen: .set-File-Generierung, Backtest-Runner-Steuerung, numerische Parameter-Mathematik und AI-Score-Parsing. Sie haben die höchste Implementierungspriorität (Business Risk: KRITISCH).

Test-ID	Testfall	Vorbedingung	Testschritte (Given/When/Then)	Testdaten	Erw. Ergebnis
TC-ENG-01	**SetFile_ValidParams**	Dateisystem schreibbar, Gültige EA-Parameter-Map vorhanden	Given: Vollständige Parameter-Map (LotSize=0.1, SL=50, TP=100) When: generateSetFile() wird aufgerufen Then: Datei existiert, Encoding ist UTF-16LE mit BOM, alle Schlüssel-Wert-Paare vorhanden	LotSize||0.1||N StopLoss||50||N TakeProfit||100||N	Datei erstellt, 3 Zeilen, UTF-16LE BOM verifiziert
TC-ENG-02	**SetFile_MissingParams**	Parameter-Map ohne Pflichtfeld "LotSize"	Given: Parameter-Map ohne LotSize-Eintrag When: generateSetFile() wird aufgerufen Then: IllegalArgumentException mit Meldung "Required parameter missing: LotSize"	Map: {StopLoss=50}	IllegalArgumentException geworfen, keine Datei erstellt
TC-ENG-03	**SetFile_InvalidPath**	Zielverzeichnis /readonly/ mit chmod 444	Given: Schreibgeschütztes Zielverzeichnis When: generateSetFile() aufgerufen Then: IOException wird abgefangen, Fehler geloggt (WARN Level), kein Crash	Path: /readonly/ea.set	IOException geloggt, Methode liefert false
TC-ENG-04	**SetFile_SpecialChars**	Parameter mit Sonderzeichen im Namen	Given: Parametername "EA_Name" mit Wert "Stratégie Ä" When: Datei wird generiert Then: Sonderzeichen korrekt in UTF-16LE kodiert und beim Einlesen identisch	EA_Name=Stratégie Ä	Round-Trip: Gelesen == Geschrieben
TC-ENG-05	**SetFile_EmptyConfig**	Leere Parameter-Map übergeben	Given: Leere HashMap When: generateSetFile() aufgerufen Then: Fehlermeldung geloggt, Methode liefert false, keine Datei wird angelegt	Map: {}	false returned, no file created, WARN logged
TC-ENG-06	**SetFile_LinuxPathing**	OS: Linux (sys.prop mocked)	Given: OS=Linux simuliert When: Pfad "C:\MT5\EA.ex5" wird formatiert Then: Slashes werden zu Forward-Slashes normalisiert	Input: C:\MT5\EA.ex5	Output: C:/MT5/EA.ex5
TC-ENG-07	**SetFile_WindowsPathing**	OS: Windows (sys.prop mocked)	Given: OS=Windows simuliert When: Forward-Slash-Pfad wird formatiert Then: Forward-Slashes werden zu Backslashes normalisiert	Input: C:/MT5/EA.ex5	Output: C:\MT5\EA.ex5
TC-ENG-08	**SetFile_Overwriting**	Bereits vorhandene .set Datei	Given: Datei mit Inhalt "LotSize=0.5" existiert When: generateSetFile() mit LotSize=0.1 aufgerufen Then: Datei enthält exakt die neuen Daten, kein Append, kein doppelter Eintrag	Existing: LotSize=0.5 New: LotSize=0.1	File contains only LotSize=0.1

Test-ID	Testfall	Vorbedingung	Testschritte (Given/When/Then)	Testdaten	Erw. Ergebnis
TC-ENG-09	**Runner_StandardExecution**	MT5 ProcessBuilder gemockt, Simulierter Erfolgs-Report vorhanden	Given: ProcessBuilder liefert Exit-Code 0 When: runBacktest() aufgerufen Then: Status == SUCCESS, BacktestResult-Objekt populiert	Mock Report: report_success.xml	BacktestResult.status == SUCCESS
TC-ENG-10	**Runner_MT5Crash**	MT5 Process mockt Exit-Code -1	Given: ProcessBuilder liefert Exit-Code -1 When: runBacktest() aufgerufen Then: Status == FAILED, Fehler in Log (ERROR), kein Absturz der App	Mock exit code: -1	RunStatus.FAILED, Exception geloggt
TC-ENG-11	**Runner_Timeout**	MT5 Process simuliert keine Terminierung	Given: ProcessBuilder-Prozess terminiert nicht innerhalb 5s (Timeout konfiguriert) When: waitFor(5, SECONDS) läuft ab Then: Process wird destroyForcibly() getötet, Status == TIMEOUT	Timeout: 5000ms (Test-Config)	RunStatus.TIMEOUT, Process destroyed
TC-ENG-12	**Runner_LogParsingSuccess**	Valide MT5 Report-HTML (English locale)	Given: MT5 HTML-Report "en_success_report.htm" im test/resources When: ReportParser.parse() aufgerufen Then: NetProfit=1250.50, Drawdown=8.5%, Trades=43	File: en_success_report.htm	NetProfit=1250.50, DD=8.5%, Trades=43
TC-ENG-13	**Runner_LogParsingCorrupt**	Leere oder beschädigte Berichtsdatei	Given: Datei "corrupt_report.htm" mit HTML-Fehler (fehlende Tabellen) When: parse() aufgerufen Then: ParseException, Status PARSE_ERROR, keine NPE	File: corrupt_report.htm (0 bytes)	ParseException, no NullPointerException
TC-ENG-14	**Runner_ZeroTrades**	Gültiger Report, aber 0 ausgeführte Trades	Given: Report mit "Total Trades: 0" When: parse() aufgerufen Then: result.trades == 0, result.status == NO_TRADES	TotalTrades: 0 in report	Result marked NO_TRADES
TC-ENG-15	**Runner_NegativeProfit**	Report mit negativem Gesamtergebnis	Given: Report "Net Profit: -520.00" When: parse() aufgerufen Then: result.netProfit == -520.00 (nicht absolut), isLoss == true	NetProfit: -520.00	netProfit=-520.00, isLoss=true
TC-ENG-16	**Runner_MarginCall**	Report enthält "Margin Call" Eintrag	Given: Report mit Margin-Call-Event in Trade-History When: parse() aufgerufen Then: result.hasMarginCall == true, Pass als ungültig markiert	Trade: MarginCall event present	hasMarginCall=true, status=INVALID
TC-ENG-17	**Math_PermutationCount**	Gültige Start/Step/Stop-Werte	Given: Start=10, Stop=20, Step=2 When: calculatePassCount() aufgerufen Then: Ergebnis == 6 (10,12,14,16,18,20)	Start=10, Stop=20, Step=2	PassCount = 6
TC-ENG-18	**Math_InvalidStep**	Step-Wert = 0	Given: Start=10, Stop=20, Step=0 When: calculatePassCount() aufgerufen Then: ArithmeticException mit Meldung "Step must be != 0"	Step = 0	ArithmeticException thrown
TC-ENG-19	**Math_NegativeStep**	Negativer Step bei Start > Stop	Given: Start=20, Stop=10, Step=-2 When: calculatePassCount() aufgerufen Then: Ergebnis == 6 (20,18,16,14,12,10)	Start=20, Stop=10, Step=-2	PassCount = 6
TC-ENG-20	**Math_FloatingPointPrecision**	Fließkomma-Akkumulation (Lotsize)	Given: Start=0.01, Step=0.01, Stop=0.10 When: generateValues() aufgerufen Then: Kein Wert wie 0.020000001, alle exakt auf 2 Dezimalstellen	BigDecimal-Sequenz 0.01..0.10	No floating-point drift, exact values

Test-ID	Testfall	Vorbedingung	Testschritte (Given/When/Then)	Testdaten	Erw. Ergebnis
TC-ENG-21	**Math_MinMaxInverted**	Start > Stop bei positivem Step	Given: Start=20, Stop=10, Step=+2 When: validateRange() aufgerufen Then: ValidationException "Start must be <= Stop for positive step"	Start=20, Stop=10, Step=+2	ValidationException thrown
TC-ENG-22	**AI_ScoreExtraction_Valid**	LLM-Antwort mit gültigem Score	Given: LLM-Response "Analysis complete. Score: 85/100. Strategy shows plateau..." When: AIScoreParser.extract() aufgerufen Then: score == 85 (Integer)	"Score: 85/100"	Extracted score = 85
TC-ENG-23	**AI_ScoreExtraction_Invalid**	LLM-Antwort ohne erkennbaren Score	Given: LLM-Response "Unable to analyze strategy at this time." When: extract() aufgerufen Then: score == 0 (Fallback-Wert), kein Exception	"Unable to analyze..."	score = 0 (fallback)
TC-ENG-24	**AI_PromptGeneration**	Vollständige RobustnessResult-Daten vorhanden	Given: RobustnessResult mit Drawdown=12.5%, Trades=150, CV%=8.3 When: buildPrompt() aufgerufen Then: Prompt enthält alle 3 Platzhalter {drawdown}, {trades}, {cv}	DD=12.5%, Trades=150, CV=8.3	All placeholders filled in prompt
TC-ENG-25	**AI_NetworkError**	OpenRouter API nicht erreichbar (Timeout)	Given: HTTP-Client wirft SocketTimeoutException nach 3s When: callOpenRouter() aufgerufen (3 Retries konfiguriert) Then: 3 Retry-Versuche, danach AIServiceException, Score=0	Mock: SocketTimeout x3	3 retries, AIServiceException, score=0

5.2 Database Management (TC-DB-01 bis TC-DB-16)

Diese 16 Tests sichern die gesamte Persistenzschicht ab. Das Hauptaugenmerk liegt auf Thread-Sicherheit, Transaktionsintegrität und korrektem SQL-Verhalten unter Fehler- und Lastbedingungen. Basis ist eine H2 In-Memory DB.

Test-ID	Testfall	Vorbedingung	Testschritte (Given/When/Then)	Testdaten	Erw. Ergebnis
TC-DB-01	**DB_PoolInit**	HikariCP konfiguriert mit validen H2-Credentials	Given: HikariConfig mit url=jdbc:h2:mem:testdb When: DatabaseManager.init() aufgerufen Then: Connection-Pool aktiv, getConnection() liefert valide Connection	url: jdbc:h2:mem:testdb	Pool active, connection != null
TC-DB-02	**DB_InvalidCredentials**	Falsche JDBC-URL oder Passwort	Given: HikariConfig mit falscher URL "jdbc:h2:mem:???" When: init() aufgerufen Then: SQLException, App-Start bricht kontrolliert ab, Fehlermeldung in Log	url: jdbc:h2:mem:???	SQLException, controlled shutdown
TC-DB-03	**DB_SchemaMigration**	Frische, leere H2 In-Memory DB	Given: Leere DB ohne Tabellen When: initializeSchema() aufgerufen Then: Alle 3 Tabellen existieren: BACKTEST_RUNS, OPTIMIZATION_RESULTS, EA_SAVED_CONFIGS	Empty H2 DB	All 3 tables created and queryable
TC-DB-04	**DB_GracefulShutdown**	Laufender Connection Pool	Given: Pool mit 3 aktiven Connections When: DatabaseManager.close() aufgerufen Then: Pool shutdown, alle Connections freigegeben, kein Memory Leak (via JVM-Profiler)	Pool size: 3 connections	Pool closed, no active connections
TC-DB-05	**CRUD_InsertValid**	Initialisiertes Schema, gültiges Objekt	Given: Vollständiges OptimizationResult-Objekt (alle Pflichtfelder gesetzt) When: dao.insert(result) aufgerufen Then: SELECT COUNT(*) == 1, alle Felder identisch zum ursprünglichen Objekt	OptimizationResult(profit=500, dd=10%)	COUNT=1, fields match exactly
TC-DB-06	**CRUD_InsertDuplicate**	Datensatz mit ID=1 bereits in DB	Given: Datensatz ID=1 existiert When: dao.insert() mit identischer ID aufgerufen Then: Kein Exception-Crash; stattdessen UPDATE ausgeführt (Upsert-Logik)	Duplicate ID: 1	Upsert executed, COUNT remains 1
TC-DB-07	**CRUD_ReadById**	Datensatz mit ID=42 in DB vorhanden	Given: DB enthält Datensatz ID=42, Profit=750.0 When: dao.findById(42) aufgerufen Then: Zurückgegebenes Objekt: id==42, profit==750.0, keine anderen Datensätze	DB row: ID=42, Profit=750.0	Object with id=42, profit=750.0 returned
TC-DB-08	**CRUD_ReadAll**	5 Datensätze für Strategy "RSI_EA" in DB	Given: 5 Einträge für Strategy="RSI_EA" und 3 für "MA_EA" When: dao.findByStrategy("RSI_EA") aufgerufen Then: Exakt 5 Objekte zurückgegeben, keine MA_EA Einträge enthalten	5x RSI_EA, 3x MA_EA in DB	List.size() == 5, all RSI_EA
TC-DB-09	**CRUD_UpdateScore**	Datensatz ID=10 mit aiScore=0	Given: Datensatz ID=10, aiScore=0 When: dao.updateAiScore(10, 87) aufgerufen Then: SELECT aiScore WHERE id=10 liefert 87	ID=10, old score=0, new score=87	aiScore updated to 87

Test-ID	Testfall	Vorbedingung	Testschritte (Given/When/Then)	Testdaten	Erw. Ergebnis
TC-DB-10	**CRUD_DeleteSelected**	6 Datensätze in DB mit IDs 1-6	Given: IDs [1,2,3,4,5,6] in DB When: dao.deleteByIds([2,4]) aufgerufen Then: COUNT==4, IDs [1,3,5,6] vorhanden, IDs [2,4] nicht mehr auffindbar	Delete IDs: [2, 4]	IDs 2,4 gone; 1,3,5,6 remain
TC-DB-11	**CRUD_DeleteAll**	10 Datensätze in Tabelle	Given: 10 Einträge in OPTIMIZATION_RESULTS When: dao.deleteAll() aufgerufen Then: COUNT==0, Tabellenstruktur (Spalten) weiterhin vorhanden	10 rows in table	COUNT=0, schema intact
TC-DB-12	**Bulk_InsertPerformance**	H2 In-Memory DB, 10.000 Objekte	Given: Liste mit 10.000 OptimizationResult-Objekten When: dao.insertBatch(list) mit Transaktion aufgerufen Then: Alle 10.000 Einträge vorhanden AND Gesamtdauer < 2000ms	10,000 objects, batch transaction	All inserted AND time < 2s
TC-DB-13	**Bulk_RollbackOnError**	100 Objekte; Objekt Nr. 50 hat NULL-Pflichtfeld	Given: 100 Objekte, Nr. 50: strategy=NULL (NOT NULL constraint) When: insertBatch() aufgerufen Then: Transaction vollständig gerollt; COUNT==0 (nicht 49)	100 objects, #50 has null field	Rollback: COUNT=0, no partial data
TC-DB-14	**Query_BestProfit**	5 Datensätze mit unterschiedlichen Profits	Given: Profits [200, -50, 1500, 750, 320] When: dao.findTopByProfit(3) aufgerufen Then: Liste [1500, 750, 320] in genau dieser Reihenfolge	Profits: 200,-50,1500,750,320	Ordered: [1500, 750, 320]
TC-DB-15	**Query_LowestDrawdown**	5 Datensätze mit unterschiedlichen Drawdowns	Given: Drawdowns [15%, 5%, 22%, 8%, 3%] When: dao.findByLowestDrawdown(3) aufgerufen Then: Liste [3%, 5%, 8%] in genau dieser Reihenfolge	DDs: 15,5,22,8,3 %	Ordered: [3%, 5%, 8%]
TC-DB-16	**Query_ComplexFilter**	Gemischte Datensätze: einige erfüllen beide Kriterien	Given: 10 Einträge mit verschiedenen Profit/DD-Kombinationen When: dao.findByFilter(profit>1000, dd<10%) aufgerufen Then: Exakt nur Datensätze zurück, die BEIDE Bedingungen erfüllen	Filter: profit>1000 AND dd<10%	Only rows matching both criteria

5.3 Dukascopy API & Tick-Data (TC-DUK-01 bis TC-DUK-18)

Diese 18 Tests sichern den gesamten Datenakquisitions-Stack ab: Authentifizierung, Download-Logik mit Error-Recovery, BI5-LZMA-Dekompression und CSV-Transformation. Netzwerk-Isolation via WireMock ist zwingend erforderlich.

Test-ID	Testfall	Vorbedingung	Testschritte (Given/When/Then)	Testdaten	Erw. Ergebnis
TC-DUK-01	**Auth_LoginSuccess**	WireMock: POST /login → 200 OK + Session-Token	Given: WireMock liefert {token: "abc123"} auf POST /login When: client.login("user","pass") aufgerufen Then: client.sessionToken == "abc123", Folgerequest enthält Authorization-Header	Response: {token: "abc123"}	Token stored, used in next request
TC-DUK-02	**Auth_LoginFailed**	WireMock: POST /login → 401 Unauthorized	Given: WireMock liefert HTTP 401 When: login() aufgerufen Then: AuthenticationException, UI erhält Fehlermeldung "Login fehlgeschlagen"	HTTP 401 Response	AuthException, UI notified
TC-DUK-03	**Auth_TokenExpired**	Erster Request: 401. Re-Login: 200.	Given: Erster Download-Request liefert 401; Re-Login liefert neuen Token When: downloadTicks() aufgerufen Then: Auto-Re-Login ausgeführt, Download wird mit neuem Token fortgesetzt	1st: 401, Re-login: 200+token	Download continues after re-auth
TC-DUK-04	**Auth_RateLimit**	WireMock: HTTP 429 für 3 Requests, dann 200	Given: Erste 3 Requests → 429, 4. Request → 200 When: downloadDay() aufgerufen Then: Exponential Backoff (100ms, 200ms, 400ms), 4. Versuch erfolgreich	3x 429, then 200	Retry with backoff, success on 4th
TC-DUK-05	**Auth_NoConnection**	WireMock off, kein Server	Given: Kein Server verfügbar (Connection refused) When: login() aufgerufen Then: NetworkException sofort (kein Hang), UI-Fehlermeldung "Keine Verbindung"	No server running	NetworkException, no hang
TC-DUK-06	**DL_SingleDay**	WireMock: Gültige .bi5 Bytes für 1 Tag	Given: WireMock liefert valide BI5-Binärdaten für EURUSD 2024-01-15 When: downloadDay("EURUSD", 2024-01-15) aufgerufen Then: Datei eurusd_20240115.bi5 existiert im Cache, Größe > 0 Bytes	Mock BI5 for EURUSD 2024-01-15	File cached, size > 0
TC-DUK-07	**DL_ChunkingMultiYear**	Request: 5 Jahre Daten	Given: Download-Anfrage: 2019-01-01 bis 2024-01-01 When: downloadRange() aufgerufen Then: WireMock empfängt exakt 60 separate Monats-Requests (5 Jahre * 12 Monate)	2019-01-01 to 2024-01-01	Exactly 60 monthly requests made
TC-DUK-08	**DL_ResumePartial**	Bereits 6 von 12 Monate im Cache	Given: Jan-Jun 2024 bereits im Datei-Cache When: downloadYear("EURUSD", 2024) aufgerufen Then: Nur 6 Requests an WireMock (Jul-Dez), keine Duplikate	6 months cached already	Only 6 new requests sent
TC-DUK-09	**DL_InvalidSymbol**	Nicht-existierendes Symbol "XXX"	Given: WireMock liefert 404 auf Request für "XXX" When: downloadDay("XXX", ...) aufgerufen Then: InvalidSymbolException, WARN-Log "Symbol not found: XXX", kein Crash	Symbol: "XXX", HTTP 404	InvalidSymbolException, WARN logged

Test-ID	Testfall	Vorbedingung	Testschritte (Given/When/Then)	Testdaten	Erw. Ergebnis
TC-DUK-10	**DL_DiskFull**	Festplatte voll (IOException beim Schreiben)	Given: FileOutputStream wirft IOException "No space left on device" When: downloadDay() schreibt Datei Then: Download gestoppt, DiskFullException, keine korrupten Teil-Dateien	IOException: no space	DiskFullException, no partial file
TC-DUK-11	**DL_ChecksumVerify**	Server liefert Datei mit falschem Checksum	Given: WireMock liefert BI5-Daten, Checksum-Header stimmt nicht mit Inhalt überein When: verifyAndCache() aufgerufen Then: ChecksumException, Datei gelöscht, erneuter Download-Request	Checksum mismatch in header	File deleted, retry triggered
TC-DUK-12	**DL_WeekendHandling**	Download-Anfrage für Samstag 2024-01-20	Given: Request für Samstag When: downloadDay("EURUSD", 2024-01-20) aufgerufen Then: Null/leere Response ignoriert, kein Eintrag in Cache, kein Error-Log	Date: 2024-01-20 (Saturday)	No error, day silently skipped
TC-DUK-13	**CSV_ParseValid**	Standard-konforme Dukascopy CSV im Speicher	Given: CSV-String "20240115 000000123,1.09500,1.09502,0,0" When: CsvParser.parse() aufgerufen Then: TickData-Objekt: timestamp=20240115-000000.123, bid=1.09500, ask=1.09502	"20240115 000000123,1.09500,1.09502,0,0"	TickData: bid=1.09500, ask=1.09502
TC-DUK-14	**CSV_ParseInvalidHeader**	CSV mit falschem Header "Date,Ask,Bid"	Given: CSV-Header "Date,Ask,Bid" statt "Timestamp,Bid,Ask,BidVol,AskVol" When: parse() aufgerufen Then: IncompatibleFormatException, Parser lehnt Datei ab	Header: "Date,Ask,Bid"	IncompatibleFormatException thrown
TC-DUK-15	**CSV_ParseCorruptRow**	CSV mit einer Zeile ohne Ask-Preis	Given: 10 valide Zeilen + 1 fehlerhafte Zeile "20240115 000001000,1.09500" When: parseAll() aufgerufen Then: 10 TickData-Objekte, Fehler-Zähler == 1, kein Absturz	10 valid + 1 corrupt row	10 objects, errorCount=1, no crash
TC-DUK-16	**CSV_TransformToMT5**	Dukascopy UTC-Timestamps in CSV	Given: Dukascopy-CSV mit UTC+0 Timestamps When: transform(timezone="Europe/Berlin") aufgerufen Then: Output-CSV hat UTC+1 Timestamps (Winterzeit) / UTC+2 (Sommerzeit)	UTC 2024-01-15 12:00 → Berlin	Output: 2024-01-15 13:00 CET
TC-DUK-17	**CSV_GapDetection**	CSV mit 5-Stunden Lücke unter der Woche	Given: Tick-Daten 08:00-09:00 und dann 14:00-15:00 (5h Lücke) When: analyzeGaps() aufgerufen Then: GapWarning-Objekt mit gap_start=09:00, gap_end=14:00, duration=5h	Gap: 09:00 to 14:00 (5h)	GapWarning with correct times
TC-DUK-18	**CSV_EmptyFile**	CSV-Datei mit 0 Bytes	Given: Leere CSV-Datei (0 Bytes) When: parse() aufgerufen Then: Leere Liste zurückgegeben, kein NullPointerException, INFO-Log "Empty file skipped"	File: 0 bytes	Empty list, no NPE, INFO logged

5.4 UI Models & Reporting (TC-UI-01 bis TC-UI-14)

Diese 14 Tests validieren die Swing TableModel-Logik, Sortierfunktionen und Export-Prozesse. Swing-Rendering-Tests werden bewusst ausgeschlossen; im Fokus steht die reine Geschäftslogik der Modelle und der Export-Generatoren.

Test-ID	Testfall	Vorbedingung	Testschritte (Given/When/Then)	Testdaten	Erw. Ergebnis
TC-UI-01	**Model_UpdateAsync**	EDT leer, Background-Thread bereit	Given: BacktestResult kommt von Worker-Thread When: model.addResult(result) aufgerufen Then: SwingUtilities.invokeLater() wurde exakt 1x aufgerufen (Mockito verify); Tabelle zeigt neuen Eintrag	Result from worker thread	invokeLater called 1x, row added
TC-UI-02	**Model_ClearData**	5 Einträge im Modell vorhanden	Given: TableModel enthält 5 Datensätze When: model.clear() aufgerufen Then: getRowCount() == 0, fireTableDataChanged() wurde aufgerufen	Model has 5 rows	getRowCount()==0, event fired
TC-UI-03	**Model_RemoveSelected**	Modell mit 6 Einträgen (Index 0-5)	Given: 6 Einträge [A,B,C,D,E,F] im Modell When: removeRows([1,3]) aufgerufen (B und D) Then: Modell enthält exakt [A,C,E,F], getRowCount()==4	Remove indices [1,3]	Remaining: [A,C,E,F], size=4
TC-UI-04	**Model_PassColumnSync**	Pass #42 in Optimization gespeichert	Given: Pass #42 in OptimizerTableModel vorhanden When: SensitivityTableModel mit Pass #42 populiert Then: sensitivity.getPassId(0) == 42 identisch zu optimizer.getPassId(rowOf42)	PassId = 42 in both models	Both models: passId == 42
TC-UI-05	**Model_FormatCurrency**	Lokale: DE (Germany)	Given: Wert 1000.5 (double), Locale.GERMANY When: CurrencyFormatter.format(1000.5, "EUR") aufgerufen Then: Output == "1.000,50 €"	Value: 1000.5, Locale: DE	Output: "1.000,50 EUR"
TC-UI-06	**Sort_ByScoreAsc**	Modell mit AI-Scores [85, 20, 95, 60]	Given: 4 Zeilen mit Scores [85, 20, 95, 60] When: sort(column="aiScore", ASC) aufgerufen Then: Reihenfolge == [20, 60, 85, 95]	Scores: [85,20,95,60]	Sorted: [20,60,85,95]
TC-UI-07	**Sort_ByScoreDesc**	Modell mit AI-Scores [85, 20, 95, 60]	Given: 4 Zeilen mit Scores [85, 20, 95, 60] When: sort(column="aiScore", DESC) aufgerufen Then: Reihenfolge == [95, 85, 60, 20]	Scores: [85,20,95,60]	Sorted: [95,85,60,20]
TC-UI-08	**Sort_HandleNulls**	Modell mit gemischten Scores und null	Given: Scores [85, null, 60, null, 20] When: sort(ASC) aufgerufen Then: [20, 60, 85, null, null] – Nulls ans Ende	Scores: [85,null,60,null,20]	Nulls at end: [20,60,85,null,null]
TC-UI-09	**Export_HTMLSuccess**	3 OptimizationResult-Objekte im Modell	Given: Modell mit 3 vollständigen Ergebnissen When: HtmlExporter.export(model, "/tmp/report.html") aufgerufen Then: Datei existiert, enthält -Tag, alle 3 Ergebnisse als -Elemente	3 results in model	HTML file with table and 3 rows
TC-UI-10	**Export_HTMLEmpty**	Leeres Modell (0 Einträge)	Given: Leeres TableModel When: export() aufgerufen Then: ExportWarningException "No data to export", keine leere HTML-Datei erstellt	Empty model, 0 rows	ExportWarningException, no file

Test-ID	Testfall	Vorbedingung	Testschritte (Given/When/Then)	Testdaten	Erw. Ergebnis
TC-UI-11	**Export_PDFSuccess**	Modell mit 5 Ergebnissen und Equity-Chart	Given: 5 Ergebnisse + Chart-Objekt vorhanden When: PdfExporter.export(model, chart, "/tmp/report.pdf") aufgerufen Then: .pdf existiert, Dateigröße > 5KB, lesbar (via pypdf-Check)	5 results + chart data	PDF exists, size > 5KB, valid
TC-UI-12	**Export_PDFPermissions**	Zielordner /readonly/ mit chmod 444	Given: Schreibgeschützter Ausgabepfad When: export() aufgerufen Then: PermissionException, UI-Dialog "Zugriff verweigert – Ordner nicht beschreibbar"	Path: /readonly/ (chmod 444)	PermissionException, UI dialog shown
TC-UI-13	**Chart_DataSeriesAdd**	Leeres ChartModel	Given: Leeres EquityChartModel When: addDataPoint(timestamp=T, balance=1000.0) aufgerufen Then: dataPoints.size()==1, repaintRequested==true	Add: T=1000ms, balance=1000.0	size=1, repaintRequested=true
TC-UI-14	**Chart_DataSeriesMax**	Chart-Kapazität: max 10.000 Punkte	Given: Chart-Kapazität konfiguriert auf 10.000 When: 15.000 Punkte hinzugefügt Then: dataPoints.size() <= 10.000 (Downsampling aktiv) OR älteste Punkte verworfen	15,000 data points added	Points <= 10,000, no OutOfMemory

6. Qualitätsmetriken & Erfolgskriterien

Die folgenden Metriken sind verbindliche Qualitätstore (Quality Gates), die vor jeder Produktionsfreigabe automatisiert in der CI-Pipeline geprüft werden. Unterschreitungen blockieren den Merge automatisch.

Tabelle 5: Qualitätsgates & Metriken

Metrik	Gate-Wert	Paket / Scope	Messung
Line Coverage	> 85%	com.backtester.engine.* com.backtester.report.*	JaCoCo (CI Breaker)
Branch Coverage	> 75%	Alle Geschäftslogik-Klassen	JaCoCo Report
Gesamte Test-Laufzeit	< 15 Sekunden	Alle 73 Unit-Tests	GitHub Actions Log
Fehlerhafte Tests	0 (ZERO)	Alle Test-Suites	GitHub Actions (PR-Block)
Kritische Bugs nach Release	0 (ZERO)	Produktionsbetrieb	Bug-Tracker Monitoring
Kein "Thread.sleep" in Tests	Pflicht	Alle Test-Klassen	SonarQube Lint Rule

7. Definition of Done (DoD)

Ein Unit-Test gilt als "Done" und freigabefähig, wenn **alle** folgenden Kriterien ohne Ausnahme erfüllt sind:

1	**Code Quality**: Test ist nach Given/When/Then-Struktur dokumentiert. Keine "Magic Numbers" – alle Werte sind als benannte Konstanten deklariert.
2	**Pass Rate 100%**: Test läuft auf **Windows 10/11** lokal und in der **Linux Ubuntu 22 CI-Pipeline** zu 100% erfolgreich durch.
3	**Isolation garantiert**: Alle externen Dienste (DB, Netzwerk, Dateisystem) sind vollständig per Mockito entkoppelt. Kein `Thread.sleep()` in Testmethoden.
4	**Coverage erfüllt**: JaCoCo-Bericht bestätigt vollständigen Durchlauf der Zielmethode (Branch & Line Coverage).
5	**Peer-Review bestanden**: Code wurde durch 4-Augen-Prinzip geprüft und im GitHub Pull Request abgenickt.
6	**CI Pipeline grün**: GitHub Actions Build läuft durch, kein roter Check auf dem Branch.

8. Aufwandsabschätzung & Ressourcenplanung

Die Kalkulation basiert auf 1 Personentag (PT) = 8 Arbeitsstunden eines erfahrenen Java-Entwicklers (Senior Level, min. 3 Jahre JUnit/Mockito-Erfahrung). Die Schätzungen berücksichtigen ausschließlich die Implementierung gemäß dieser Spezifikation – kein Refactoring-Overhead von Legacy-Code.

Tabelle 6: Aufwandsabschätzung nach Projektphasen

Phase	Komponente	Tests	Aufwand (PT)	Haupttätigkeiten
1	Setup & Infrastruktur	—	1.5 PT	JUnit5/Mockito/JaCoCo in pom.xml; abstrakte BaseTest-Klassen; H2 Config; WireMock Setup
2	Core Engine Tests	25 Tests	3.5 PT	Aufwendiges FS-Mocking für Set-Files; async ProcessBuilder-Simulation; Floating-Point-Tests
3	Database Management	16 Tests	2.0 PT	H2 Schema-Setup per @BeforeEach; DAO-Test-Fixtures; Transaktions/Rollback-Szenarien
4	Dukascopy API & Ticks	18 Tests	3.0 PT	WireMock-Scenarios; BI5-IO-Stream-Tests; CSV-Parser Randfälle; Timezone-Konvertierung
5	UI Models & Reporting	14 Tests	1.5 PT	TableModel-Logik; Sortiertests; Exporter-Verifikation (HTML/PDF); Chart-Kapazität
6	CI/CD Integration	—	0.5 PT	GitHub Actions Workflow; JaCoCo-Report als Artifact; PR-Blockierung bei Fehler
7	Code Review & Fixes	—	1.5 PT	Puffer für Reviews, Nachbesserungen, Refactoring schlecht testbarer Klassen
GESAMT		**73 Tests**	**13.5 PT**	**Ca. 2.7 Entwickler-Wochen (1 Vollzeit-Entwickler)**

9. Risikomanagement

Tabelle 7: Risikoregister

Risiko-ID	Risikobezeichnung	Wahrsch.	Auswirkung	Risiko-Level	Gegenmaßnahme
R-01	Refactoring-Bedarf bei stark gekoppelten Klassen (z.B. RobustnessRunner, static-Methoden)	MITTEL	HOCH	HOCH	Dependency Injection vorab einführen. Aufwand in Phase 7 (1.5 PT) berücksichtigt.
R-02	Test-Laufzeit > 15 Sekunden durch echte Thread.sleep() oder ungemockte IO-Operationen	MITTEL	MITTEL	MITTEL	CI-Build-Schritt mit Timeout-Limit. SonarQube-Regel verbietet Thread.sleep in Tests.
R-03	MT5-Report-Format ändert sich durch MT5-Update (HTML-Struktur, Localization)	NIEDRIG	HOCH	MITTEL	Dedizierter "ReportFormat-Monitor"-Test. Parser mit Versionsdetektion ausstatten.
R-04	Dukascopy API-Änderungen brechen den DukascopyDownloader (Endpoint, Authentifizierung)	NIEDRIG	HOCH	MITTEL	WireMock-Tests sind API-unabhängig. Alerting via Integration-Test auf Staging.
R-05	Fehlende Testdaten für Edge-Cases (z.B. seltene BI5-Formate, Leap-Year-Bugs)	HOCH	MITTEL	HOCH	Strukturierte Testdaten-Bibliothek in test/resources anlegen. Dokumentiert und versioniert.
R-06	Wissenskonzentration: Nur 1 Entwickler kennt die gesamte Test-Architektur	MITTEL	HOCH	HOCH	Pairing/Reviews, Living Documentation im Confluence, kein Single Point of Failure.

10. Rollen & Verantwortlichkeiten

Tabelle 8: RACI-Matrix (R=Responsible, A=Accountable, C=Consulted, I=Informed)

Rolle	Person / Team	Verantwortlichkeiten	RACI
Lead QA Engineer	Thomas Nickel	Erstellung & Pflege des Testkonzepts; Qualitätskontrolle aller Tests; Entscheidung über Freigabe	R
Java Developer (Senior)	Entwicklungsteam	Implementierung der Unit-Tests gemäß Spezifikation; Code-Review; Refactoring für Testbarkeit	A
DevOps Engineer	Infrastruktur-Team	CI/CD-Pipeline-Setup (GitHub Actions); JaCoCo-Integration; Merge-Blocker-Konfiguration	C
Geschäftsführung	Management	Bereitstellung von Ressourcen (13.5 PT); Abnahme des Konzepts; Priorisierungsentscheidungen	I
Product Owner	Fachbereich Trading	Fachliche Validierung der Testfälle; Priorisierung nach Business Risk	C

Dokumentenhistorie
Version 1.0 – Initiale Erstellung (01. Mai 2026)
Version 1.1 – Detaillierung der Testfälle, Diagramme ergänzt (07. Mai 2026)

Genehmigung
Erstellt: Thomas Nickel
Status: Zur Management-Freigabe