КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Лабораторна робота №2

з дисципліни «Методи забезпечення якості програмних систем»
на тему: «Тестовий план для веб-сайту «Binance» та її API інтерфейсів»

Виконав:
Студент групи ПЗС-1
Лутай Артем Сергійович

КИЇВ 2025

Вступ

Метою даного тестового плану є визначення підходів, стратегії та умов проведення тестування торгової платформи «Binance» – високошвидкісної біржі цифрових активів з функціоналом спотової, ф'ючерсної та опціонної торгівлі, управління криптогаманцями та доступом для алгоритмічних систем через API.

Тестування проводиться з метою:

1. Перевірки відповідності реалізованої системи специфікації програмних вимог (SRS);
2. Виявлення дефектів у критично важливих компонентах: ядрі зведення ордерів (Matching Engine) та білінговій системі;
3. Підтвердження стабільності та відповідності документації публічних (Public) та приватних (Private) API інтерфейсів;
4. Оцінки надійності механізмів безпеки, захисту транзакцій та персональних даних користувачів.

У межах тестування будуть охоплені:

1. **Функціональні модулі:** торговий термінал (графіки, стакан, історія угод), управління ордерами (створення, скасування, типи ордерів), API інтерфейси (REST, WebSocket), гаманець (депозити, виведення, внутрішні перекази), система безпеки (2FA, управління ключами), адміністративна панель;
2. **Нефункціональні характеристики:** продуктивність (низька затримка/latency, пропускна здатність/TPS), надійність, безпека, доступність, сумісність та зручність використання.

Об’єкти тестування

Об’єктами тестування є основні функціональні та нефункціональні складові торгової платформи «Binance» та її програмних інтерфейсів (API), визначені у специфікації вимог.

Функціональні модулі

1. **Торговий термінал (Web/Mobile)** — візуалізація ринкових даних (графіки TradingView), відображення книги ордерів (Order Book) у реальному часі, стрічка останніх угод.
2. **Управління ордерами** — створення, редагування та скасування ордерів різних типів (Limit, Market, Stop-Limit, OCO, Trailing Stop); валідація вхідних параметрів (ціна, кількість, крок ціни).
3. **Matching Engine (Ядро)** — алгоритми зведення ордерів за принципом Price-Time Priority, часткове виконання ордерів (Partial Fill), розрахунок комісій (Maker/Taker).
4. **Public API (REST)** — ендпоінти для отримання публічних ринкових даних: тікери, історичні свічки (Klines), глибина стакану (Depth), час сервера.
5. **Private API (Signed)** — захищені ендпоінти для торгівлі та управління акаунтом; механізм підпису запитів (HMAC SHA256), керування лімітами запитів (Rate Limits).
6. **WebSocket Stream** — підсистема реального часу для трансляції змін у стакані (Diff. Depth), агрегованих угод та оновлень статусів ордерів користувача (UserDataStream).
7. **Гаманець (Wallet)** — генерація депозитних адрес для різних мереж, логіка зарахування депозитів, обробка заявок на виведення коштів (Withdrawal), внутрішні перекази.
8. **Система безпеки та автентифікації** — вхід у систему, налаштування 2FA (Google Auth, YubiKey), управління API-ключами (створення, права доступу, білий список IP).
9. **KYC Верифікація** — завантаження та обробка документів користувача, перевірка Liveness (селфі), зміна рівнів лімітів на виведення.

10. **Адміністративна панель** — управління торговими парами (лістинг/делістинг), моніторинг підозрілих транзакцій, керування блокуванням акаунтів.

Нефункціональні характеристики

1. **Продуктивність (Performance)** — час відгуку API (Latency), пропускна здатність Matching Engine (TPS), швидкість доставки даних через WebSocket.
2. **Надійність і доступність (Reliability & Availability)** — аптайм системи (SLA 99.9%), коректне відновлення сесій після обриву з'єднання, цілісність даних при збоях.
3. **Безпека (Security)** — стійкість до Replay-атак, захист від DDoS, шифрування даних, захист від SQL Injection та XSS.
4. **Сумісність (Compatibility)** — коректна робота клієнтських бібліотек (Python, Java, C++) з API, підтримка сучасних браузерів.
5. **Зручність використання (Usability)** — зрозумілість повідомлень про помилки API, зручність навігації у веб-інтерфейсі.
6. **Масштабованість (Scalability)** — здатність системи обробляти різке зростання кількості запитів під час високої волатильності ринку.

Підходи до тестування

Тестування платформи «Binance» буде проводитися з використанням комбінації функціональних і нефункціональних технік, з сильним акцентом на автоматизацію перевірок API та навантажувальне тестування ядра системи.

Функціональне тестування

1. **Метод «чорної скриньки» (black-box testing):** Тестування логіки роботи біржі через інтерфейси (UI та API) без доступу до вихідного коду Matching Engine.
2. **Позитивні сценарії (happy path):** Перевірка основних бізнес-процесів — реєстрація, генерація депозитної адреси, розміщення лімітного ордера, виконання угоди (Trade), списання комісії, виведення коштів.
3. **Негативні сценарії:** Перевірка обробки помилок — спроба торгівлі з недостатнім балансом, введення ціни з неправильним кроком (tick size), використання простроченого timestamp у запиті API, спроба виведення коштів без 2FA.
4. **API Тестування:** Валідація JSON-структури відповідей, перевірка відповідності HTTP статус-кодів (200 OK, 400 Bad Request, 401 Unauthorized, 429 Too Many Requests), перевірка механізму підпису запитів (HMAC SHA256).
5. **Регресійне тестування:** Перевірка критичного функціоналу (розрахунок балансів, логін) після кожного оновлення бекенду.

Нефункціональне тестування

1. **Продуктивність (Performance):** Вимірювання затримок (Latency) при розміщенні ордерів, перевірка пропускної здатності (TPS) Matching Engine та швидкості доставки даних через WebSocket (Market Data).
2. **Навантажувальне тестування (Load Testing):** Емуляція пікових навантажень (тисячі запитів на секунду) для перевірки стабільності API Gateway та черги повідомлень.
3. **Безпека (Security):** Тестування на вразливості OWASP Top 10, перевірка стійкості до Replay-атак, перевірка rate-limiting

(захист від DDoS та спаму запитами), аудит шифрування даних.

4. **Сумісність (Compatibility):** Тестування веб-терміналу в основних браузерах (Chrome, Firefox, Safari) та перевірка роботи API з популярними клієнтськими бібліотеками (CCXT, Python binance-connector).
5. **Юзабіліті (Usability):** Оцінка зрозумілості інтерфейсу для трейдера, читабельності графіків та інформативності повідомлень про помилки в API.

Автоматизоване тестування (Automated Testing)

Автоматизація є пріоритетною для API та бекенд-логіки. Автотести покриватимуть:

1. **API сценарії:** Повний цикл торгівлі (Place -> Query -> Cancel), перевірка балансів.
2. **WebSocket сценарії:** Підписка на потоки, перевірка отримання ping/pong фреймів, відновлення з'єднання.
3. **Складні сценарії:** Order Matching (перевірка, що ордер виконався саме за тією ціною, яка була вказана).

Інструменти:

1. Pytest + Requests: Для функціонального тестування REST API.
2. Websocket-client: Для тестування стрімів реального часу.
3. Selenium / Playwright: Для автоматизації UI сценаріїв (Login, KYC flow).
4. JMeter / K6: Для навантажувального тестування.

Автотести інтегруються в CI/CD пайплайн (GitLab CI / Jenkins) і запускаються при кожному коміті/злитті у вказану гілку.

Інтеграційне тестування (Integration Testing)

Інтеграційне тестування спрямоване на перевірку взаємодії між компонентами платформи та зовнішніми системами:

1. Ethereum, перевірка коректності відображення депозитів на балансі користувача.

2. **Інтеграція з KYC-провайдером:** Перевірка процесу відправки документів та отримання callback-відповіді про статус верифікації (Approved/Rejected).
3. **Інтеграція з сервісами 2FA:** Валідація одноразових паролів (Google Authenticator, YubiKey).
4. **Взаємодія внутрішніх модулів:** Синхронізація даних між Matching Engine (ядром) та базою даних користувачів (баланси).

Ролі та ресурси

У процесі тестування платформи «Binance» задіяні такі спеціалісти з відповідними зонами відповідальності:

QA Automation Engineer (Інженер з автоматизації)

1. розробляє фреймворк для автоматизованого тестування API (REST/WebSocket) та навантажувального тестування;
2. створює скрипти для генерації синтетичного ринкового трафіку;
3. налаштовує запуск автотестів у CI/CD пайплайнах.

Manual QA Engineer (Тестувальник ручного тестування)

1. виконує тестування користувацького інтерфейсу (Web та Mobile), перевіряє локалізацію та зручність (Usability);
2. тестує складні бізнес-сценарії, які важко автоматизувати (KYC-верифікація, апеляції P2P, відновлення 2FA);
3. документує знайдені дефекти та перевіряє їх виправлення.

Backend Developer (C++ / Go)

1. здійснює Unit-тестування компонентів ядра (Matching Engine);
2. виправляє дефекти в логіці обробки ордерів та розрахунку балансів;
3. надає технічну підтримку QA-команді при аналізі логів продуктивності.

Security Specialist (Спеціаліст з безпеки)

1. проводить аудит смарт-контрактів та API на вразливості (Penetration Testing);
2. перевіряє надійність механізмів шифрування та зберігання ключів;
3. тестує систему на стійкість до DDoS-атак та маніпуляцій із запитами.

DevOps / SRE (Site Reliability Engineer)

1. розгортає та підтримує тестові середовища (Staging/Testnet);
2. налаштовує системи моніторингу (Prometheus/Grafana) для відстеження метрик під час навантажувальних тестів;
3. керує тестовими нодами блокчейнів.

Product Owner (Власник продукту)

1. затверджує тестові сценарії та критерії приймання (Acceptance Criteria);

2. пріоритезує дефекти залежно від їх впливу на бізнес та безпеку коштів.

У процесі тестування, також будуть задіяні наступні ресурси:

Технічні ресурси:

1. **Тестове середовище (Testnet):** Ізольований кластер серверів, що емулює продуктивну інфраструктуру (Matching Engine, API Gateway, DB), але використовує віртуальні активи.
2. **Блокчейн-ноди:** Підключення до тестових мереж (Bitcoin Testnet, Ethereum Sepolia/Goerli) для валідації депозитів та виводів без реальних фінансових витрат.
3. **Генератори навантаження:** Виділені сервериз високою пропускною здатністю для запуску навантажувальних скриптів (емуляція тисяч клієнтів).
4. **Мобільні ферми:** Набір реальних пристроїв (iOS/Android) для тестування мобільного додатку.

Програмні ресурси:

1. **Управління тестуванням:** Jira / TestRail (планування, трекінг дефектів, звітність).
2. Інструменти автоматизації: *Python (Pytest, Requests, Websocket-client)* — для тестування API. *Selenium / Playwright* — для UI тестування веб-терміналу.
3. **Інструменти навантаження:** K6 / JMeter / Gatling (для тестування пропускної здатності API).
4. **Робота з API:** Postman / Insomnia (для ручних запитів та налагодження).
5. Моніторинг та аналіз: *Wireshark* — аналіз мережевих пакетів та WebSocket фреймів. *Grafana / Kibana* — візуалізація логів та метрик сервера. *Burp Suite* — сканування вразливостей веб-додатку.

Людські ресурси (для лабораторного проекту):

1. 1 Lead QA / Test Architect (планування та стратегія).
2. 1 QA Automation Engineer (API та навантаження).
3. 1 Manual QA (UI та сценарії користувача).

4. 2 Backend Developers (підтримка та фікси).
5. 2 Frontend Developers (підтримка та фікси).

Ризики

Під час тестування торгової платформи «Binance» можливі такі ризики, що можуть вплинути на терміни, якість або повноту виконання робіт:

Організаційні ризики

1. **Зміни в регуляторних вимогах:** Раптова зміна правил KYC/AML або фінансового моніторингу (наприклад, вимоги MiCA або SEC), що призведе до необхідності термінової зміни бізнес-логіки та переробки тестової документації.
2. **Зміна пріоритетів бізнесу:** Висока волатильність ринку може змусити команду змістити фокус з планових задач на термінові фікси або впровадження нових торгових пар, що скоротить час на тестування основного функціоналу.
3. **Комунікаційні бар'єри:** Недостатня комунікація між командою розробки ядра (Matching Engine) та QA-командою щодо специфіки нових алгоритмів зведення ордерів.

Технічні ризики

1. **Нестабільність тестових мереж (Testnets):** Можливі збої, хард-форки або зупинка генерації блоків у тестових мережах блокчейнів (Bitcoin Testnet, Ethereum Goerli), що заблокує тестування депозитів та виводів.
2. **Проблеми синхронізації часу:** Розбіжність системного часу між серверами та клієнтськими машинами автотестів, що призведе до масових помилок валідації підпису API (Timestamp for this request is outside of the recvWindow).
3. **Нестабільність сторонніх API:** Збої у роботі зовнішніх провайдерів KYC (перевірка документів) або платіжних шлюзів, які неможливо контролювати.
4. **Обмеження тестового середовища:** Тестовий стенд може не витримати навантаження, аналогічного до реального (Production), що ускладнить точну оцінку продуктивності (Latency).

Ресурсні ризики

1. **Брак вузькопрофільних спеціалістів:** Відсутність у команді тестувальників з глибоким розумінням специфіки HFT (високочастотної торгівлі) та блокчейн-технологій.
2. **Висока вартість інфраструктури:** Обмежений бюджет на оренду потужних серверів для проведення повномасштабного навантажувального тестування (Load Testing).
3. **Затримки у наданні доступів:** Довгий процес отримання доступів до захищених сегментів (гарячі/холодні гаманці) з міркувань безпеки.

Ризики якості

1. **Фінансові втрати через помилки округлення:** Ризик пропуску дефектів, пов'язаних з втратою точності (floating point errors) при розрахунках комісій або частковому виконанні ордерів.
2. **Вразливості безпеки:** Виявлення критичних дірок у безпеці (наприклад, можливість Replay-атаки) на пізніх етапах, що вимагатиме архітектурних змін.
3. **Деградація продуктивності:** Ризик того, що після оновлення час відгуку API (Latency) перевищить допустимі межі, що є критичним для алгоритмічних трейдерів.