**Subject: Updated Full Test Plan for MMF Tokenization with Fireblocks and Hedera/Canton Blockchain Integration**

**1. Immediate Steps: What We Will Test**

Here’s how I’ll approach the project practically:

**A. Understanding Key Components of the System**

1. **Fireblocks Integration**: This is the wallet solution.
   * **What to test**: Wallet creation, token deposits, transfers, and withdrawal.
   * **How**: Ensure that every transaction from wallet creation to MMF token purchases and transfers works.
2. **Hedera/Canton Blockchain**: The DLT platform for recording transactions.
   * **What to test**: Ensure that the MMF tokens are correctly minted on the blockchain.
   * **How**: Verify that tokens are issued, recorded, and immutable on the blockchain.
3. **ERC-3643 Smart Contract**: The smart contract for tokenization.
   * **What to test**: Token minting, transfers, and access control.
   * **How**: Ensure smart contracts behave as expected (mint tokens, distribute to wallets, restrict unauthorized actions).

**B. Testing Phases (from my practical standpoint)**

**Phase 1: Unit Testing (Development Level)**

* **Smart Contracts**:
  + Test each function of the **ERC-3643** smart contract to ensure the issuance and transfer of tokens.
  + **Tools**: Hardhat or Truffle.
  + **Action**: Write unit tests for minting functions, ownership checks, and token balance updates.
* **Fireblocks Wallet API**:
  + Validate wallet creation, transaction handling, and funding via Fireblocks API.
  + **Tools**: **Bruno** (for API testing).
  + **Action**: Test wallet creation, balance checks, and transactions using predefined scenarios (valid/invalid wallet, incorrect funds).

**Phase 2: Integration Testing**

* **Integration of Fireblocks + Hedera/Canton Blockchain + ERC-3643**:
  + Test if tokens minted from Fireblocks are properly reflected on the Hedera/Canton blockchain.
  + **Tools**: **Bruno** for API testing, **Cucumber** with **Playwright** for UI testing.
  + **Action**: Integrate smart contracts and wallet API, test end-to-end functionality.
  + **Expected Outcome**: Transaction from Fireblocks (wallet) to Hedera should result in a successful token creation and confirmation on the blockchain.

**C. Automation Plan (100% Automation Strategy)**

To achieve **100% test automation**, here's how we will do it:

**1. Tool Selection: Bruno, JMeter, Playwright + Cucumber**

* **Bruno**:
  + **API Testing Tool** that will be used to test Fireblocks wallet APIs and the token minting process via the ERC-3643 smart contracts.
  + It will allow us to automate testing of wallet creation, transaction requests, token transfer validations, etc.
* **JMeter**:
  + **Performance Testing Tool** that will simulate heavy load and transactions to verify how well the system can handle high volumes of users and transactions (especially critical for token issuance and transfers).
  + **Action**: Set up JMeter to simulate thousands of token minting and transfer requests to monitor system performance under load.
* **Playwright + Cucumber**:
  + **UI Testing Tool** that will allow us to automate end-to-end UI tests for user journeys, from wallet creation to MMF token purchases.
  + **Playwright**: Used for interacting with the front-end (UI automation).
  + **Cucumber**: Used for behavior-driven testing, writing readable and executable user stories (BDD).
  + **Action**: Automate UI testing for:
    - User registration.
    - Wallet creation via Fireblocks.
    - Token purchase from the bank platform.
    - Transaction verification on Hedera blockchain.

**2. Steps to Automate**

* **API Testing (Bruno)**:
  + Automate tests for the wallet API (Fireblocks API) and ERC-3643 smart contract interaction via Bruno.
  + Create automated scripts to validate the following:
    - Wallet creation (valid and invalid scenarios).
    - Funds deposit and withdrawal from wallet.
    - Token minting and transfer actions.
* **UI Testing (Playwright + Cucumber)**:
  + Automate the following user journeys:
    - Register and log into the platform.
    - Create a wallet and fund it using Fireblocks.
    - Buy MMF tokens.
    - Verify token minting on Hedera blockchain and update the user interface accordingly.
  + **Scenario**:
    - **Given**: The user is on the homepage.
    - **When**: The user registers and logs into the platform, creates a wallet via Fireblocks, funds the wallet, and purchases MMF tokens.
    - **Then**: The system should mint tokens and display the updated balance.
  + **Action**: Run these tests continuously during the development cycle to ensure that every part of the UI works as expected.
* **Performance Testing (JMeter)**:
  + Set up **JMeter** to simulate thousands of token minting and transfer requests under high load.
  + **Action**: Conduct stress testing on the Fireblocks API, smart contract calls, and blockchain interactions to verify that the platform can handle a large volume of simultaneous transactions without failures or performance degradation.
  + Monitor response times and server load during peak usage.

**3. Self-Healing Tests**

* **Bruno** and **Playwright** (with Cucumber) will have the capability to handle minor changes in the APIs and UI, reducing the need for constant test script updates. The goal is to reduce the overhead of test maintenance by leveraging AI-powered capabilities for test adaption.

**D. Reporting and Metrics**

**1. Continuous Integration**

* Tests will be part of our **CI pipeline** (e.g., Jenkins).
* Every time code is pushed to the repository:
  + Automated tests will run for **API**, **UI**, and **Performance** tests.
  + Reports will show test results in **real-time**, detailing which steps failed and why.

**2. Monitoring Test Coverage**

* **Test Coverage**: I’ll ensure that 100% of code paths (smart contracts, wallet interactions, transactions) are tested.
* **Reporting Tools**: Use **SonarQube** to measure test coverage and ensure no critical parts of the system are untested.

**3. Performance Monitoring**

* We will integrate **performance testing** for token transaction speed using **JMeter**, making sure that our system can handle high loads.
  + **Expected Load**: We’ll simulate thousands of transactions to ensure scalability.

**E. Practical Execution Plan and Timeline**

1. **Week 1-2**:
   * Write unit tests for smart contracts.
   * Set up Bruno for API testing of Fireblocks and smart contracts.
   * Begin automating the token purchase UI flow using **Playwright + Cucumber**.
2. **Week 3-4**:
   * Complete integration tests with Fireblocks and Hedera using Bruno and Playwright.
   * Automate full end-to-end user journey (wallet creation, token purchase, blockchain validation).
   * Set up **JMeter** for performance testing.
3. **Week 5-6**:
   * Finalize automated testing on all components (smart contracts, wallets, token transfers).
   * Set up CI/CD pipeline for automated test execution.
   * Final review and performance load testing with **JMeter**.

**Conclusion**

This approach will **achieve 100% test automation** with **Bruno** for **API automation**, **Playwright + Cucumber** for **UI automation**, and **JMeter** for **performance testing**. The integration of **AI-powered self-healing** tests will further reduce manual test maintenance, ensuring the system remains robust as we move through different phases of development.