SMOKE Suite:

For smoke testing, the goal is to quickly validate that the core functionality of each API works as expected. Below are detailed test cases for each API, focusing on basic functionality, common error scenarios, and general service checks.

1. Bank Accounts API

Test Case 1: Get All Accounts

- **Description**: Verify that the endpoint retrieves all bank accounts associated with the user.
- **Method**: GET
- **URL**: /api/v1/bank-accounts
- **Expected Outcome**:
    - Status Code: 200 OK
    - Response Body: List of bank accounts with account_id, account_type, bank_name, balance, and currency.
- **Test Steps**:

1. Send a GET request to /api/v1/bank-accounts.
2. Verify that the status code is 200 OK.
3. Check that the response body contains valid account data (non-empty list, correct data types).

Test Case 2: Get Account Details

- **Description**: Verify that the endpoint retrieves details for a specific bank account.
- **Method**: GET
- **URL**: /api/v1/bank-accounts/{account_id}
- **Expected Outcome**:
    - Status Code: 200 OK
    - Response Body: Details of the specified bank account including transactions.
- **Test Steps**:

1. Send a GET request to /api/v1/bank-accounts/{account_id}.
2. Verify that the status code is 200 OK.
3. Ensure the response body matches the expected structure and includes transaction history.

Test Case 3: Account Not Found

- **Description**: Ensure the endpoint returns an appropriate error when the specified account ID does not exist.
- **Method**: GET
- **URL**: /api/v1/bank-accounts/nonexistent_account_id
- **Expected Outcome**:
    - Status Code: 404 Not Found
    - Response Body: Error message indicating that the account was not found.
- **Test Steps**:

1. Send a GET request to /api/v1/bank-accounts/nonexistent_account_id.
2. Verify that the status code is 404 Not Found.
3. Confirm that the response body includes an error message.

2. Investments API

Test Case 4: Get Investment Portfolio

- **Description**: Verify that the endpoint retrieves a summary of all investments.
- **Method**: GET
- **URL**: /api/v1/investments
- **Expected Outcome**:
    - Status Code: 200 OK
    - Response Body: List of investments with investment_id, type, name, quantity, current_price, etc.
- **Test Steps**:

1. Send a GET request to /api/v1/investments.
2. Verify that the status code is 200 OK.
3. Ensure the response body contains valid investment data.

Test Case 5: Get Investment Details

- **Description**: Verify that the endpoint retrieves detailed information for a specific investment.
- **Method**: GET
- **URL**: /api/v1/investments/{investment_id}

- **Expected Outcome**:
    - Status Code: 200 OK
    - Response Body: Detailed information including transaction_history.
- **Test Steps**:

1. Send a GET request to /api/v1/investments/{investment_id}.
2. Verify that the status code is 200 OK.
3. Ensure the response body includes transaction history and matches the expected data structure.

Test Case 6: Investment Not Found

- **Description**: Ensure the endpoint returns an appropriate error when an investment ID does not exist.
- **Method**: GET
- **URL**: /api/v1/investments/nonexistent_investment_id
- **Expected Outcome**:
    - Status Code: 404 Not Found
    - Response Body: Error message indicating that the investment was not found.
- **Test Steps**:

1. Send a GET request to /api/v1/investments/nonexistent_investment_id.
2. Verify that the status code is 404 Not Found.
3. Confirm that the response body includes an error message.

3. EPF Balance API

Test Case 7: Get EPF Balance

- **Description**: Verify that the endpoint retrieves the current EPF balance for a specific user.
- **Method**: GET
- **URL**: /api/v1/epf/{user_id}
- **Expected Outcome**:
    - Status Code: 200 OK
    - Response Body: user_id, epf_balance, last_updated, and currency.
- **Test Steps**:

1. Send a GET request to /api/v1/epf/{user_id}.
2. Verify that the status code is 200 OK.
3. Ensure the response body contains the expected data fields.

Test Case 8: EPF Balance Not Found

- **Description**: Ensure the endpoint returns an appropriate error when the user ID does not exist.
- **Method**: GET
- **URL**: /api/v1/epf/nonexistent_user_id
- **Expected Outcome**:
    - Status Code: 404 Not Found
    - Response Body: Error message indicating that the EPF balance was not found.
- **Test Steps**:

1. Send a GET request to /api/v1/epf/nonexistent_user_id.
2. Verify that the status code is 404 Not Found.
3. Confirm that the response body includes an error message.

4. Liabilities API (Loans)

Test Case 9: Get All Liabilities

- **Description**: Verify that the endpoint retrieves a list of all loans and liabilities.
- **Method**: GET
- **URL**: /api/v1/liabilities
- **Expected Outcome**:
    - Status Code: 200 OK
    - Response Body: List of liabilities including loan_id, type, bank_name, amount_due, remaining_term, etc.
- **Test Steps**:

1. Send a GET request to /api/v1/liabilities.
2. Verify that the status code is 200 OK.
3. Ensure the response body contains valid liability data.

Test Case 10: Get Loan Details

- **Description**: Verify that the endpoint retrieves detailed information for a specific loan.

- **Method**: GET
- **URL**: /api/v1/liabilities/{loan_id}
- **Expected Outcome**:
    - Status Code: 200 OK
    - Response Body: Detailed information including monthly_installment, transaction_history.
- **Test Steps**:

1. Send a GET request to /api/v1/liabilities/{loan_id}.
2. Verify that the status code is 200 OK.
3. Confirm that the response body includes transaction history and matches the expected data structure.

Test Case 11: Loan Not Found

- **Description**: Ensure the endpoint returns an appropriate error when the loan ID does not exist.
- **Method**: GET
- **URL**: /api/v1/liabilities/nonexistent_loan_id
- **Expected Outcome**:
    - Status Code: 404 Not Found
    - Response Body: Error message indicating that the loan was not found.
- **Test Steps**:

1. Send a GET request to /api/v1/liabilities/nonexistent_loan_id.
2. Verify that the status code is 404 Not Found.
3. Confirm that the response body includes an error message.

1. Authentication API

Test Case 1: Generate Access Token

- **Description**: Verify that the endpoint generates a valid access token.
- **Method**: POST
- **URL**: /auth/token
- **Request Body**:

- {
- "client_id": "valid_client_id",
- "client_secret": "valid_client_secret",
- "grant_type": "client_credentials"
- }
- **Expected Outcome**:
    - Status Code: 200 OK
    - Response Body: Includes access_token, token_type, and expires_in.

Test Case 2: Invalid Credentials

- **Description**: Verify the response for invalid client credentials.
- **Request Body**: Invalid client_id or client_secret.
- **Expected Outcome**:
    - Status Code: 401 Unauthorized
    - Response Body: Error message indicating invalid credentials.

---

2. Get Spending Insights API

Test Case 3: Retrieve Spending Insights

- **Description**: Verify that spending insights are retrieved correctly.
- **Method**: GET
- **URL**: /spending-insights
- **Headers**:
    - Authorization: Bearer {valid_access_token}
- **Query Parameters**:
    - Valid start_date and end_date.
- **Expected Outcome**:
    - Status Code: 200 OK
    - Response Body: Includes total_spent, top_categories, and monthly_trends.

Test Case 4: Missing Authorization Header

- **Description**: Verify response when the Authorization header is missing.

- **Expected Outcome**:
    - Status Code: 401 Unauthorized
    - Response Body: Error message about missing authorization.

Test Case 5: Invalid Date Range

- **Description**: Verify response for an invalid start_date or end_date.
- **Expected Outcome**:
    - Status Code: 400 Bad Request
    - Response Body: Error message specifying invalid date format or range.

Notes for Implementation

- **Automation Tools**: Use Postman for quick validation and Rest Assured or JMeter for performance testing.
- **Assertion Focus**:
    - Response structure (fields and types).
    - Correct status codes and error handling.
- **Environment**:
    - Test in staging with production-like configurations.
    - Use synthetic data for edge-case simulations.
- **Execution Frequency**: Run this suite before every major deployment to ensure system readiness.
- **Create CI/CD pipeline to run this suite**.