Project Documentation

1. Introduction

The Project is a sophisticated software solution designed to perform statistical analyses and calculations on financial data sourced from the National Bank of Poland's API, available at http://api.nbp.pl/. This system is tailored for detailed insights into currency trends and statistics over various time periods.

2. System Architecture

System Architecture for this project is based on desktop applications utilizing PySide framework for Python. Unit tests performed with the pytests library. Data requests to NBP API are made with the request python module. While matplotlib module was used to create currency charts. Project designed only for devices with Windows operating systems. Project can be built as executable file (.exe extension).

3. Used Technologies

Python with modules:

- ***requests*** version 2.31.0: used for creating and handling NBP API requests
- ***pytest*** version 7.4.0: used for testing project via unit tests
- ***numpy*** version 1.26.4: used for various mathematical operation on currency data
- ***freezegun*** version 1.1.0: support module used for testing application
- ***PySide6*** version 6.7.1: framework used to build GUI
- ***matplotlib*** version 3.9.0: used for creating charts
- ***pyinstaller*** version 6.8.0:used for creating executable files

4. Project Structure

Project is divided into following modules:

APP FOLDER

A. *api.py*

Contains core project functionality such as function for getting changes distribution or loading currency data from NBP API

Module methods:

A1. get_sessions_data

Description: Retrieves the number of growth, decline, and unchanged sessions for a specified currency over a given analysis period.

Args:

- currency (str): The currency code for which session data is to be retrieved.
- analysisPeriod (AnalysisPeriod): The period for which the session data is to be analyzed.

Returns:

- tuple: A tuple containing three integers representing the number of growth sessions, decline sessions, and unchanged sessions.

Raises:

- ValueError: If an invalid analysis period is provided.
- requests.RequestException: If there is an issue connecting to the NBP API.

A2. get_statistical_measures

Description: Calculates statistical measures (median, mode, standard deviation, and coefficient of variation) for a specified currency over a given analysis period.

Args:

- currency (str): The currency code for which statistical measures are to be calculated.
- analysisPeriod (AnalysisPeriod): The period for which statistical measures are to be calculated.

Returns:

- tuple: A tuple containing four float values representing the median, mode, standard deviation, and coefficient of variation.

Raises:

- ValueError: If an invalid analysis period is provided.
- requests.RequestException: If there is an issue connecting to the NBP API.

A3. get_changes_distribution

Description: Analyzes the distribution of changes in value between two currencies over a specified period.

Args:

- currency_1 (str): The currency code for the first currency.
- currency_2 (str): The currency code for the second currency.
- start_date (date): The start date for analyzing the changes.
- analysisPeriod (AnalysisPeriod): The period for which the analysis of changes is to be performed.

Returns:

- tuple: A tuple containing two lists: the first representing the histogram values for each bin, and the second representing bin boundaries.

Raises:

- ValueError: If the start date is in the future or an invalid analysis period is provided.
- requests.RequestException: If there is an issue connecting to the NBP API.

B. *app_ui.ui* and ***app_ui.py***

First file define layout of application interface. The second one is automatically generated by QT User Interface compiler (Pyside framework) based on app_ui.ui file.

C. *constants.py*

Contains enumerate class AnalysisPeriod with following values:
WEEK = 1,
TWO_WEEKS = 2,
MONTH = 3,
QUARTER = 4,
HALF_YEAR = 5,
YEAR = 6

D. *main.py*

Main module of the project, allowing launching application from IDE. Can be used for creating executables files.

TESTS FOLDER

E. *test_calculate_statistical_measures.py*

test_invalid_currency()

This test verifies that the get_statistical_measures function raises a ValueError when an invalid currency code is provided.
The function should raise a ValueError with the message "Invalid request parameters".

test_no_currency()

This test verifies that the get_statistical_measures function raises a ValueError when an empty currency code is provided.
The function should raise a ValueError with the message "Invalid request parameters".

test_invalid_analysis_period()

This test verifies that the get_statistical_measures function raises a ValueError when an invalid analysis period is provided.
The function should raise a ValueError with the message "Error: not a valid time period".

test_valid_requests()

This test verifies that the get_statistical_measures function returns the correct statistical measures for various currencies over different analysis periods.
The function should return the correct median, mode, standard deviation, and coefficient of variation.

test_no_internet_connection()

This test verifies that the get_statistical_measures function raises a requests.RequestException when there is no internet connection or the API is unavailable.
Simulating no internet connection using mocking.
The function should raise a requests.RequestException with the message "Connection to NBP API not available".

test_get_statistical_measures()

This test verifies that the get_statistical_measures function correctly calculates the median, mode, standard deviation, and coefficient of variation for different currencies over various analysis periods.
The function should return a tuple with four float values representing

the statistical measures for each case.

F. *test_changes_distribution.py*

test_invalid_currency_1()

Test case for handling invalid first currency input. This test verifies that the get_changes_distribution() function raises a ValueError when the first currency code is invalid.

test_invalid_currency_2()

Test case for handling invalid second currency input. This test verifies that the get_changes_distribution() function raises a ValueError when the second currency code is invalid.

test_invalid_currency_1_and_2()

Test case for handling both first and second currency inputs being invalid. This test verifies that the get_changes_distribution() function raises a ValueError when both currency codes are invalid.

test_invalid_analysis_period()

Test case for handling invalid analysis period input. This test verifies that the get_changes_distribution() function raises a ValueError when the analysis period is invalid.

test_invalid_start_date_format()

Test case for handling invalid start date input. This test verifies that the get_changes_distribution() function raises a ValueError when the start date is invalid or in the future.

test_valid_requests()

Test case for handling valid requests. This test verifies that the get_changes_distribution() function correctly processes valid inputs and returns expected histogram and bins for given currency pairs and analysis periods.

test_no_internet_connection()

Test case for handling no internet connection. This test verifies that the get_changes_distribution() function raises a requests.RequestException when there is no internet connection or the API is unavailable.

test_get_changes_distribution()

Test for determining the distribution of monthly or quarterly changes

in currency pairs. This test verifies that the get_changes_distribution() function correctly calculates the histogram and bins for changes in currency pairs over a specified period. It checks the output for both monthly and quarterly analysis periods.

G. *test_get_sessions_data.py*

test_invalid_currency()

Test case for testing function behavior with invalid currency input. This test verifies that the function raises a ValueError when an invalid currency code is provided.

test_no_currency()

Test case for testing function behavior with no currency input. This test ensures the function raises a ValueError when no currency code is provided.

test_invalid_analysis_period()

Test case for testing function behavior with invalid analysis_period. This test checks that the function raises a ValueError when an invalid analysis_period is provided.

test_valid_requests()

Test case for testing function behavior with valid input. This test verifies the function's output for various valid currency codes and AnalysisPeriod options.

test_no_internet_connection()

Test case for testing function behavior with no internet connection. This test checks that the function properly handles a scenario where the API request fails due to a network error.

test_analysis_periods()

Test case for testing different analysis periods. This comprehensive test covers the function's behavior across multiple AnalysisPeriod options, ensuring correct calculation of session data.

5. Installation Guide

A. Install Python (install at least version 3.12)
B. Install all dependencies (python libraries in .requirements.txt) using one of the following options:
- with pip install -r .requirements.txt command
- with pip install "module_name" command for every module in .requirements.txt

C. Install any IDE tools for example PyCharm Python Packages
D. Run .app/main.py using command line or IDE or create executable with following command:
- “pyinstaller --onefile --noupx --noconsole app/main.py”

6. CI/CD System

Project contains CI/CD System for automated testing and deployment of new release. Both functionalities are contained in separate files.
A. *Automated testing via file python-app.yml*
Workflow Name - name:
Exchange Rates and Statistics Generator
Trigger Conditions - on:
Specifies when the workflow should run (on push to specific branches and pull requests to those branches).
Permissions - permissions:
Ensures the workflow has necessary permissions (in this case, read access to contents).
Jobs - jobs:
Contains a single job named build.
Job Configuration - runs-on:
ubuntu-latest: Specifies the operating system for the job.
Steps:

A. **Checkout**: actions/checkout@v3 is used to fetch the repository's code.
B. **B. Setup Python**: actions/setup-python@v3 sets up Python 3.11 for the job.

C. **Install dependencies**: Installs required Python packages including flake8 and pytest, and optionally installs from requirements.txt.
D. **Lint with flake8**: Runs flake8 to lint Python code, ensuring it meets specified standards.
E. **Test with pytest**: Executes tests using pytest to validate code functionality.

B. Automated deployment via file release.yml

Workflow Name - name:

Create Release on PR Merge. Describes the purpose of the workflow, which is to automate the creation of a GitHub release upon merging a pull request into the 'release' branch.

Trigger Conditions - on:

Specifies that the workflow triggers when a pull request is merged into the 'release' branch.

Jobs - jobs:

Defines a single job named release.

Job Configuration - runs-on:

ubuntu-latest: Specifies that the job runs on the latest version of Ubuntu.

Conditional Execution - if:

Uses github.event_name and github.base_ref to ensure the job runs only when the event is a pull request merge into the 'release' branch and the merge is successful.

Steps:

A. **Checkout code**: Checks out the code of the repository.
B. **Set up Python**: Sets up the Python environment using Python version 3.11.
C. **Install dependencies**: Installs Python dependencies listed in requirements.txt.
D. **Build executable**: Uses PyInstaller to create a standalone executable from app/main.py.
E. **Create Release**: Uses actions/create-release@v1 to create a new GitHub release. It specifies the tag name, release name, and other details.

F. **Upload Release Asset**: Uses actions/upload-release-asset@v1 to upload the generated executable (main.exe) as a release asset to the GitHub release.

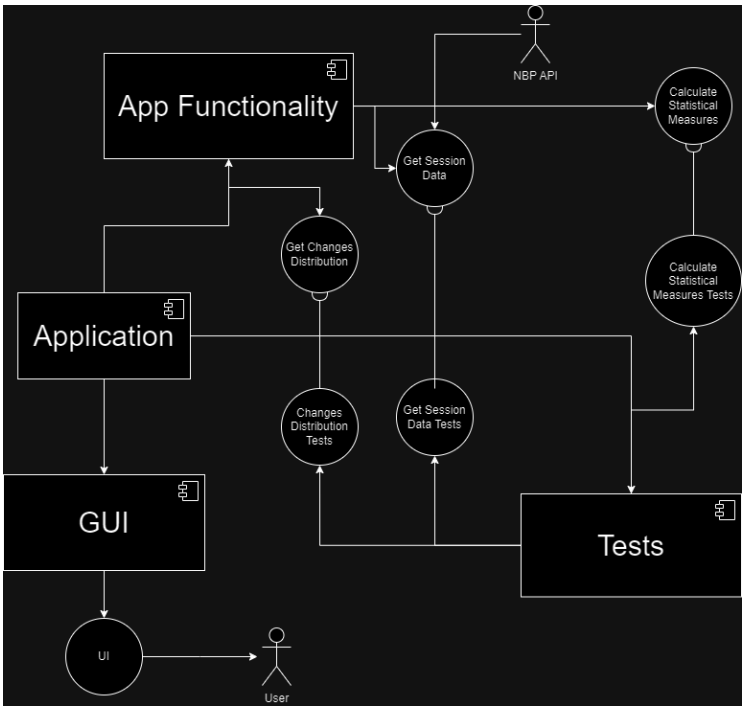
7. CI/CD Reports

CI/CD reports available on Github Action page of the project repository:
https://github.com/IIS-ZPI/ZPI2023_IO1_MVP/actions

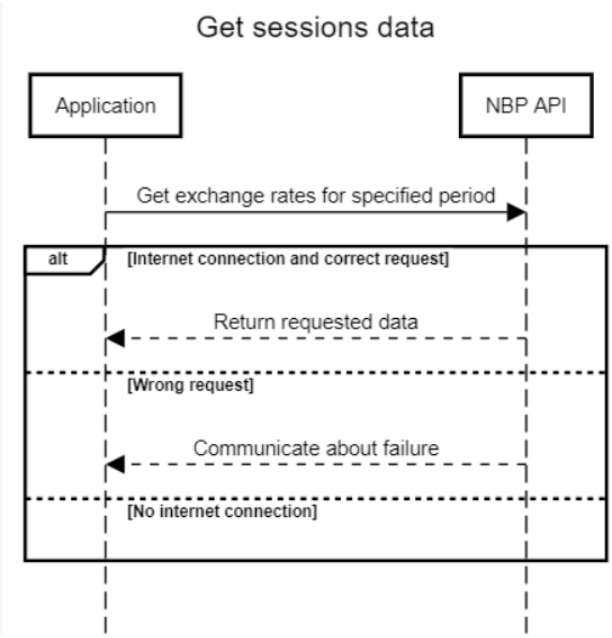
8. Project Team

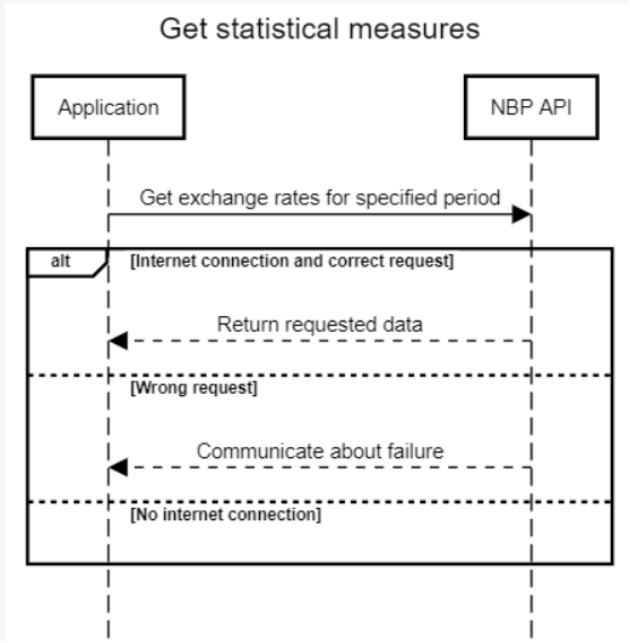
Juliusz Brodnicki 235838
Jakub Glinka 240664
Mariusz Kamiński 240696
Martyna Kaszczyk 240701
Mateusz Walkiewicz 240829
Mateusz Linard 240736

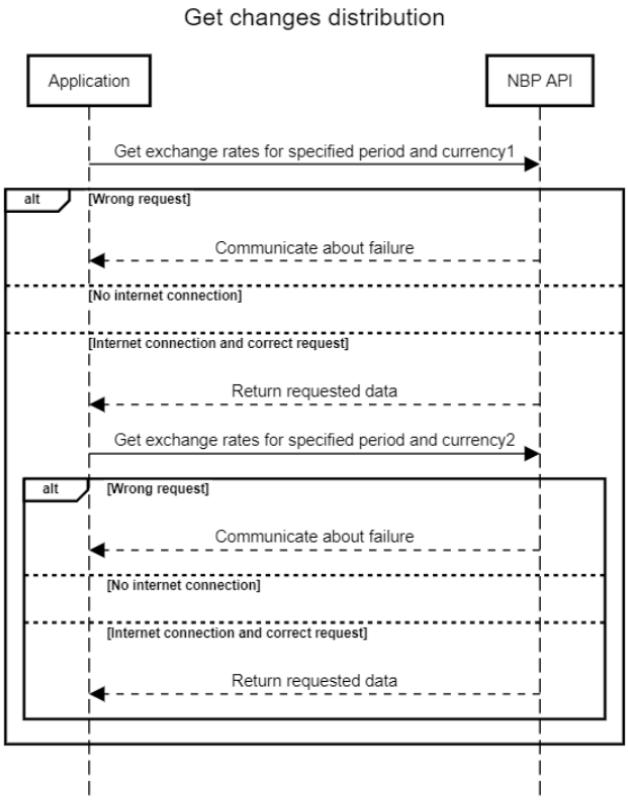
9. Component Diagram



10. Sequence Diagram







11. Activity Diagram