Configuration Manual

MSc Research Project
Programme Name

Yogesh Pant
Student ID: x21239584

School of Computing
National College of Ireland

Supervisor: Victor Del Rosal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Yogesh Pant
Student ID:	X21239584
Programme:	MSc in FinTech (MSCFTD1) **Year:** 1
Module:	Research Project
Lecturer:	Victor Del Rosal
Submission Due Date:	14-08-2023
Project Title:	Configuration Manual
Word Count:	1044 **Page Count:** 7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Yogesh Pant
Date:	14-08-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Yogesh Pant
Student ID: x21239584

1 System Specifications:

Hardware Overview:

Device name:	DESKTOP-2IQJKUO
Processor:	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
Installed RAM:	8.00 GB (7.77 GB usable)
Device ID:	F0AFB616-BCE8-4692-8C38-05400EACA0D2
Product ID:	00327-60000-00000-AA589
System type:	64-bit operating system, x64-based processor

Windows Specifications:

Edition:	Windows 11 Home Single Language
Version:	22H2
Installed on:	06-03-2023
OS build:	22621.1848
Experience:	Windows Feature Experience Pack 1000.22642.1000.0

2 Software and Other Tools:

Microsoft® Excel®

Microsoft 365 MSO (Version 2212 Build 16.0.15928.20196) 64-bit
License ID: EWW_768ebdd0-2636-4fa6-9e39-a6ce92a91ce2_728324fdd0cfcabed5

Python's PyCharm

PyCharm 2023.1.2 (Community Edition)
Build #PC-231.9011.38, built on May 16, 2023
Runtime version: 17.0.6+10-b829.9 amd64
VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
Windows 11.0
GC: G1 Young Generation, G1 Old Generation
Memory: 994M
Cores: 8

Packages present/installed in PyCharm

Package Name	Installed Version	Updated Version	Description
Faker	18.11.0	19.3.0	Generate fake data for testing and development.
Pillow	10.0.0	10.0.0	Python Imaging Library (Fork) for image processing.
contourpy	1.1.0	1.1.0	Contour plotting in Python.
cycler	0.11.0	0.11.0	Composable cycles for Matplotlib.
fonttools	4.41.0	4.42.0	Library for manipulating fonts, written in Python.
kiwisolver	1.4.4	1.4.4	Efficient C++ implementation of constraint solver.
matplotlib	3.7.2	3.7.2	Plotting library for Python.
numpy	1.25.0	1.25.2	Fundamental package for scientific computing.
packaging	23.1	23.1	Core utilities for Python software packaging.
pandas	2.0.2	2.0.3	Data manipulation and analysis library.
patsy	0.5.3	0.5.3	Descriptive statistics using formulas.
pip	22.3.1	23.2.1	Package installer for Python.
pyparsing	3.0.9	3.1.1	General parsing module for Python.
python-dateutil	2.8.2	2.8.2	Extensions to the standard datetime module.
pytz	2023.3	2023.3	World timezone definitions for Python.
scipy	1.11.1	1.11.1	Scientific library for mathematics, science, and engineering.
seaborn	0.12.2	0.12.2	Statistical data visualization based on Matplotlib.
setuptools	65.5.1	68.0.0	Easily download, build, install, upgrade, and uninstall Python packages.
six	1.16.0	1.16.0	Python 2 and 3 compatibility library.
statsmodels	0.14.0	0.14.0	Statistical modeling and econometrics library.
tzdata	2023.3	2023.3	Time zone database and utilities.
wheel	0.38.4	0.41.1	Binary package format for Python distributions.

3 Synthetic Dataset Creation:

The configuration process for generating a synthetic dataset for cross-border payment analysis involved carefully selecting attributes that capture essential transaction details. By leveraging the Python programming language and relevant libraries, we will outline the steps to construct a comprehensive dataset for investigating the impact of blockchain and distributed ledger technology (DLT) on payment efficiency.

Following libraries are required:

1. `pandas` for data manipulation and analysis.
2. `random` for generating random values.
3. `Faker` for creating fake data.
4. `math` for mathematical operations.
5. `datetime` for working with date and time.
6. `numpy` for numerical computations.

The synthetic dataset creation involves emulating cross-border payment scenarios with intricate attributes. The process involves generating attributes such as transaction specifics, currency pairs, payment intents, and global regions. The code snippet provided constructs a sample dataset comprising 10,000 rows.

```
import pandas as pd
import random
from faker import Faker
import math
import datetime
import numpy as np

# Set the number of rows in the dataset
num_rows = 10000

# Initialize the Faker library for generating fake data
fake = Faker()

# Define the related countries for each currency pair
related_countries = {
    'USD-INR': ('United States', 'India'),
    'INR-USD': ('India', 'United States'),
    'USD-GBP': ('United States', 'United Kingdom'),
    'GBP-USD': ('United Kingdom', 'United States'),
    'INR-GBP': ('India', 'United Kingdom'),
    'GBP-INR': ('United Kingdom', 'India'),
    'YUAN-INR': ('China', 'India'),
    'INR-YUAN': ('India', 'China'),
    'YUAN-USD': ('China', 'United States'),
    'USD-YUAN': ('United States', 'China'),
    'YUAN-GBP': ('China', 'United Kingdom'),
    'GBP-YUAN': ('United Kingdom', 'China')
}

# Define the common payment purposes
```

Key Steps:

1. Set the number of rows in the dataset using `num_rows`.
2. Initialize the `Faker` library for generating fake data.
3. Define related countries for each currency pair in the `related_countries` dictionary.
4. Define common payment purposes in the `payment_purposes` list.
5. Generate dummy data for each feature within a loop.
6. Append the generated data to a list of dictionaries.
7. Create a `pandas` DataFrame using the generated data.
8. Calculate additional columns such as 'Total Cost' and 'Speed of Transaction'.

4 Data Analysis, Visualization and Correlation Analysis:

Ensure that the following Python packages are installed:

1. `pandas` for data manipulation and analysis.
2. `random` for generating random values.
3. `Faker` for generating fake data.
4. `math` for mathematical operations.
5. `datetime` for working with date and time.
6. `numpy` for numerical computations.
7. `seaborn` and `matplotlib.pyplot` for visualization.

```
df.to_csv('dummy_dataset.csv', index=False)

# Define Comparison Categories: Identify the relevant categories for comparison, such as currency
comparison_categories = ['Currency Pair', 'Origin Country']

# Create an empty dictionary to store average costs and speeds by category
average_costs_by_category = {}
average_speeds_by_category = {}

# Calculate Average Cost and Speed within Categories
for comparison_category in comparison_categories:
    unique_categories = df[comparison_category].unique()

    for category in unique_categories:
        subset_data = df[df[comparison_category] == category]
        average_cost = subset_data['Total Cost'].mean()
        average_speed = subset_data['Speed of Transaction'].mean()
        average_costs_by_category[category] = average_cost
        average_speeds_by_category[category] = average_speed

# Create consolidated bar charts to compare average costs and speeds across all currency pairs
plt.figure(figsize=(10, 6))
plt.bar(average_costs_by_category.keys(), average_costs_by_category.values())
plt.xlabel('Currency Pair')
plt.ylabel('Average Cost')
plt.title('Comparison of Average Costs across all Currency Pairs')
plt.show()

plt.figure(figsize=(10, 6))
plt.bar(average_speeds_by_category.keys(), average_speeds_by_category.values())
```

The code snippet develops analysis to compare transaction costs and speeds across different categories, such as currency pairs and origin countries. Visualizations in the form of bar charts offer insights into these comparisons.

Further, the script calculates correlations among numeric attributes in the dataset and classifies them based on strength. Strong, moderate, and weak correlations are identified and printed. Additionally, a heatmap visualization of the correlation matrix is generated using `seaborn` and `matplotlib.pyplot`.

5 Cost Simulation:

Following Python packages needs to be installed:

1. `numpy` for numerical computations.
2. `pandas` for data manipulation and analysis.
3. `matplotlib.pyplot` for data visualization.

The code snippet simulates different scenarios by varying transaction attributes, including transaction amounts, fees, conversion rates, and FX charges. The simulation generates a range of possible combinations and calculates the total cost for each.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Define the simulation parameters with reduced ranges
transaction_amounts = np.arange(500, 2001, 500)
fees = np.arange(0, 11, 1)
conversion_rates = np.arange(0.8, 1.2, 0.1)
fx_charges = np.arange(0.8, 1.2, 0.1)

# Perform the simulation and calculate the total cost for each combination
results = []
for amount in transaction_amounts:
    for fee in fees:
        for rate in conversion_rates:
            for charge in fx_charges:
                total_cost = fee + (charge * amount * rate)
                results.append({
                    'Transaction Amount': amount,
                    'Fee': fee,
                    'Conversion Rate': rate,
                    'FX Charge': charge,
                    'Total Cost': total_cost
                })
```

Key Steps:

1. Define simulation parameters with reduced ranges for transaction amounts, fees, conversion rates, and FX charges.
2. Perform the simulation by iterating through the parameter combinations and calculating total costs.
3. Store the simulation results in a list of dictionaries.

Creating and Analyzing the DataFrame:

A `pandas` DataFrame is created from the simulation results, providing a structured representation of the data. Each row corresponds to a unique combination of transaction attributes and associated total cost.

Visualization:

The simulation results are visualized using separate graphs for each parameter. Matplotlib's `subplots` feature is used to arrange multiple graphs within a single figure. The visualizations offer insights into the relationship between transaction attributes and their impact on total costs.

6 Speed Simulation:

Following Python packages needs to be installed:

1. `pandas` for data manipulation and analysis.
2. `numpy` for numerical computations.
3. `matplotlib.pyplot` for data visualization.

In the given code snippet, a range of scenarios is simulated by manipulating various transaction attributes, such as transaction amounts, gas limits, transaction times, and block sizes. Through this simulation process, the transaction speed is precisely computed for each unique combination, yielding a comprehensive understanding of the interactions between these parameters.

Key Steps:

1. Define simulation parameters, such as transaction amounts, gas limits, transaction times, and block sizes.
2. Perform the simulation by iterating through parameter combinations and calculating transaction speeds.
3. Store the simulation results in a list of dictionaries.

Creating and Analyzing the DataFrame:

A `pandas` DataFrame is created from the simulation results, providing a structured representation of the data. Each row corresponds to a unique combination of parameters and the associated transaction speed.

Visualization:

The simulation results are visualized using separate graphs for each parameter. Matplotlib's `subplots` feature is used to arrange multiple graphs within a single figure. The visualizations illustrate the impact of parameter variations on transaction speed.

```
# Set the simulation parameters
transaction_amounts = [100, 500, 1000, 2000]  # Vary the transaction amounts
gas_limits = [1000, 2000, 3000, 4000]  # Vary the gas limits
transaction_times = [5, 10, 15, 20]  # Vary the transaction times in seconds
block_sizes = [1, 2, 3, 4]  # Vary the block sizes

# Create an empty list to store the simulation results
simulation_results = []

# Perform the simulation
for amount in transaction_amounts:
    for gas_limit in gas_limits:
        for time in transaction_times:
            for size in block_sizes:
                # Calculate the transaction speed based on the simulation parameters
                transaction_speed = amount / (gas_limit * (time + size))

                # Store the simulation results in a dictionary
                simulation_results.append({
                    'Transaction Amount': amount,
                    'Gas Limit': gas_limit,
                    'Transaction Time (Seconds)': time,
                    'Block Size': size,
                    'Transaction Speed': transaction_speed
                })
```