National College of Ireland

GDPR Compliance in the Metaverse - Configuration Manual

MSc Research Project
Cyber Security

Liam O'Hagan
Student ID: 22116168

School of Computing
National College of Ireland

Supervisor: Mark Monaghan

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Liam O'Hagan
Student ID:	22116168
Programme:	Cyber Security
Year:	2024
Module:	MSc Research Project
Supervisor:	Mark Monaghan
Submission Due Date:	12/08/2024
Project Title:	GDPR Compliance in the Metaverse - Configuration Manual
Word Count:	1030
Page Count:	17

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:	Liam O Hagan
Date:	11th August 2024

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):	

Contents

List of Figures

1 Introduction

1.1 Purpose

The purpose of this Configuration Manual is to provide information relating to the tools and techniques employed to gather and analyse data to develop the report "GDPR Compliance in the Metaverse"

1.2 Scope

The Configuration Manual includes any hardware and software used, manual techniques employed and details on any scripts or tools that were developed to support the processing or analysis of data.

2 System Requirements

2.1 Hardware

The Meta Quest 3 headset and hand controllers were obtained for this report.

Although this hardware was not specifically required for this report - because the report relies on data collected from the Meta Quest website and sample Apps' websites - in order to gain a better understanding and insights into the operation and use of Metaverse applications, this device did provide value.

2.2 Software

Report writing

- Overleaf Online LaTeX Editor [1]

Analysis

- VMWare Workstation Pro [2]
- Debian Linux [3]
- Sublime Text Editor [4]
- Python [5]
- RStudio [6]
- R Language [7]
- Microsoft Excel [8]
- Microsoft Word [8]

3 Data Collection

3.1 Collection of App data

No official, definitive list of Meta Quest Apps was found to be available so a manual scrape of apps was undertaken on the Meta Quest App Store [9].

Unofficial websites were discovered which contained Apps that were not available on the Quest Store. However, for the purposes of this report, only the official Meta Quest App Store [9] was considered.

The store contains five main categories of apps. Each of these contained multiple sub-pages (identified in brackets below) which contained the list of Apps for that sub-category.

- Games (36)
- Apps (6)
- Entertainment (13)
- Mixed Reality (4)
- Quest Plus (1)

The contents of each sub-category page was copied and pasted to a text file for further processing in a folder corresponding to the category, for a total of 60 text files in 5 folders.

3.2 Preparation of data

Each file contained the name of the app and other information. Some apps contains misspellings or casing errors. An app could be in multiple categories and/or subcategories so the data did require some preparation before it could be used.

The folders containing the files were copied to a Linux VM and the following Linux bash script was run against the files in each folder to clean up the data.

```
#!/bin/bash
for fileName in *.txt; do
    cleanFileName="${fileName}-clean.txt"
    grep -i -v -E "\bGet\b" "${fileName}" | grep -i -v -e "\bComing" \
    | grep -i -v -E "^€"  | grep -i -v "Pre-order" > "${cleanFileName}"
done
```

The contents of each "cleaned" file were copied into a single file for sorting and de-duplication. This file of cleaned data contained 2,197 rows of app names.

The sorting and de-duplication was carried out in the Sublime Text Editor using the following commands:

- Sort : F9
- De-duplicate : Edit / Permute / Unique

The result was a list of 653 unique Apps - 30% of its original size.

The list of apps was checked again manually to correct casing or spelling issues.

9 Apps were removed as the were demo apps and for other reasons. The apps and reasons for removal are contained in the "Removed" sheet of the "DataCollected.xlsx" workbook.

The final working list of apps numbered 644. Each of these is listed in the "Quest Store Apps" sheet in the "DataCollected.xlsx" workbook.

3.3 Identifying App Sample

As reported above, no definitive list of Apps was found. Accordingly, it was not feasible to categorise the apps for sampling without manually checking each of the 644 Apps.

Accordingly, a random selection of the Apps was chosen as the sampling method.

A set of random numbers was generated in RStudio and copied to the clipboard.

```
clipr::write_clip(runif(653))
```

This set of random numbers was pasted into the Excel workbook containing the list of Apps after sorting and de-duplication.

The list of apps was sorted against the random number list in ascending order. This random number generation and sort operation was carried out three times.

The list of random numbers and the Apps in their final order is contained in the "Quest Store Apps" sheet in the "DataCollected.xlsx" workbook.

3.4 Collection of App-related Data

For each App in the sample, the URL specific to that App was identified by searching the Meta Quest Store [9] on each App name and recording the returned URL.

Each URL was followed to view the web page for each App. Related information from each App was recorded in the "Selected" sheet of the "DataCollected.xlsx" workbook.

The data collected from each page is:

- App Name
- URL
- Cost
- Rating
- Number of ratings
- PEGI value
- In-App Purchases
- Comfort Rating
- Category
- Genre
- Version
- Release Date
- Developer
- Publisher
- Developer/App URL
- Privacy Policy URL
- Terms of Service URL
- Sensor and Device Data
  - Microphone
  - Storage
  - Location
  - Bluetooth
  - Spatial Datga
  - Hand Tracking
  - Eye Tracking
- Your Data
  - User ID
  - User Profile
  - Avatar
  - Followers
  - Usage Data
  - Age Group

3.4.1 Example App Web Page

The following is an example of an App Web Page, identifying the areas of interest in the page for the collection of data related to the app.

In the interest of viewing comfort, the image below is a composite of the page elements of interest. The actual page is longer and contains more data that has not been included in this report.

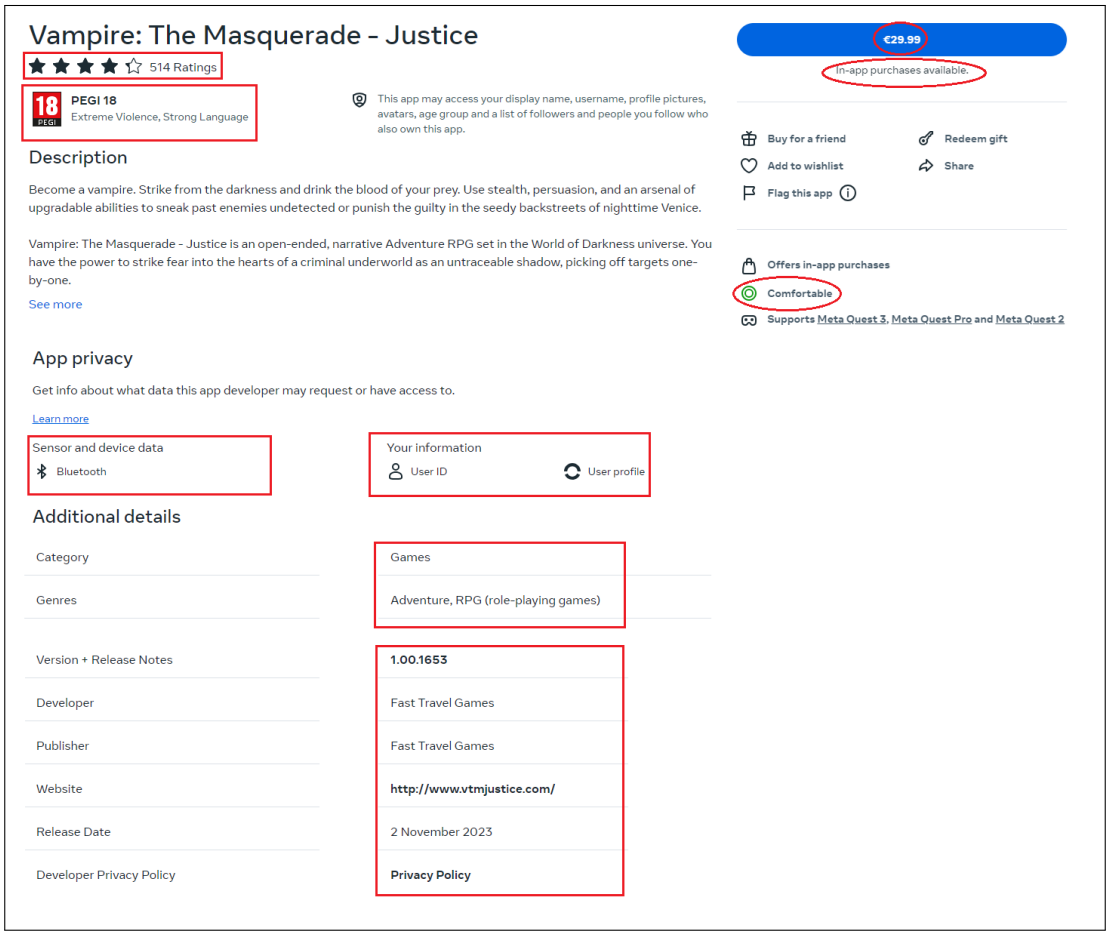

Figure 1: Example App Web Page

3.4.2 Example of device data processed

Each App's web page was examined and the data being processed was recorded in a spreadsheet.

		Sensor and Device Data					Your Information						
Index	Title	Location	Bluetooth	SpatialData	HandTracking	EyeTracking	UserID	UserProfile	Avatar	Followers	UsageData	AgeGroup	Qty
01	Vampire: The Masquerade - Justice	---	Y	---	---	---	Y	Y	---	---	---	---	3
02	Lost Recipes	---	---	---	---	---	---	---	---	---	---	---	0
03	Smash Drums	---	Y	Y	---	---	Y	Y	Y	Y	Y	---	8
04	Sniper Elite VR	---	Y	---	---	---	Y	Y	---	---	---	---	3
05	Red Matter 2	---	---	---	---	Y	Y	Y	---	---	---	---	3
06	Racket: Nx	---	---	---	---	---	Y	Y	---	Y	Y	Y	7
07	Vox Machinae	---	Y	---	---	---	Y	Y	---	---	---	---	5
08	Swords of Gargantua	---	---	---	---	---	Y	Y	---	---	---	---	3
09	Coursera	---	---	---	Y	---	---	---	---	---	---	---	1
10	Time Stall	---	---	---	---	---	Y	Y	---	---	---	---	2
11	MarineVerse Cup	---	Y	---	---	---	Y	Y	Y	Y	Y	---	8
12	The Light Brigade	---	---	---	---	Y	Y	Y	---	Y	Y	---	5
13	Blueplanet VR Explore v2	---	---	---	---	---	---	---	---	---	---	---	1
14	Paradiddle	---	---	---	---	---	Y	Y	Y	Y	---	---	6
15	Gesture VR	---	---	---	---	---	Y	Y	---	---	---	---	3
16	Beat Arena	---	---	---	---	---	Y	Y	---	---	---	---	2

Figure 2: Example Device Data

Having collected all of the data from each App's web page in relation to its data processing (and other ancillary data), a Python script was developed to generated plots to display this data in a readable way. See Section 4.1

3.5 Collection of Privacy Policy Data

Each App is required by Meta to include a link to their Privacy Policy. This link was followed for each selected App in order to evaluate the Privacy Policy.

This was an entirely manual process and it was subject to some levels of subjectivity.

3.5.1 Example of privacy policy data

Each Privacy Policy was examined and the results of each question were recorded in a spreadsheet.

		Processes				Contact			Legal							
Index	Title	Data	Clear	Accessible	Identity	Dets	Data	Purposes	Basis	Leg Int	Proc Loc	Recipients	Retention	Rights	Total	Compliance
01	Vampire: The Masquerade - Justice	Yes	2	2	2	2	2	2	2	2	2	2	2	2	24	100.0%
02	Lost Recipes	No														100.0%
03	Smash Drums	Yes	1	1	2	1	1	1	1	1	1	1	1	1	13	54.2%
04	Sniper Elite VR	Yes	1	2	2	2	2	2	1	1	1	2	2	1	19	79.2%
05	Red Matter 2	No														100.0%
06	Racket: Nx	Yes	1	1	1	1	1	1	0	0	0	1	0	0	7	29.2%
07	Vox Machinae	Yes	0	0	1	1	1	1	0	0	0	0	0	0	4	16.7%
08	Swords of Gargantua	Yes	-1	1	2	2	1	2	0	0	2	1	0	1	11	45.8%
09	Coursera	Yes	2	2	2	2	2	2	2	2	2	2	2	2	24	100.0%
10	Time Stall	Yes	2	2	2	2	2	2	2	2	2	2	2	2	24	100.0%
11	MarineVerse Cup	Yes	1	2	2	2	2	2	2	2	0	2	2	1	20	83.3%
12	The Light Brigade	Yes	0	0	1	1	0	0	0	0	0	2	0	1	5	20.8%
13	Blueplanet VR Explore v2	Yes	0	0	0	0	1	0	0	0	0	0	0	0	1	4.2%
14	Paradiddle	Yes	0	0	2	2	0	0	0	0	2	2	0	0	8	33.3%
15	Gesture VR	Yes	0	0	1	1	1	0	0	0	0	1	0	0	4	16.7%
16	Beat Arena	Yes	0	1	2	1	1	2	0	0	0	1	2	0	10	41.7%
17	Jurassic World Aftermath Collection	No														100.0%
18	Star Trek: Bridge Crew	Yes	2	2	2	2	2	2	2	2	2	2	2	2	24	100.0%

Figure 3: Example Policy Data

When all of the Privacy Policy data was collected, a Python script was developed to generate plots of the data. See section 4.2

A final Python script was developed to generate a statistical analysis of the Qty (number of data items collected) and Compliance values. See Section 4.3

4 Scripts

4.1 Plots of initial data collection

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas.api.types import CategoricalDtype
import seaborn as sns
from scipy.stats import shapiro


df = pd.read_excel('MetaAppsData.xlsx', sheet_name='Selected', skiprows=[0])


print("Creating bar chart of PEGI values")
plt.clf()
cat_type = CategoricalDtype(categories=['3', '7', '12', '16', '18', '!'], ordered=True)
df['PEGI'] = df['PEGI'].astype(str).astype(cat_type)
value_counts = df['PEGI'].value_counts().sort_index()
#value_counts = df['PEGI'].astype(str).value_counts().sort_index()
plt.bar(range(len(value_counts)), value_counts.values, tick_label=value_counts.index)
plt.title('PEGI Values')
plt.xlabel('PEGI Values')
plt.ylabel('Frequency')
plt.xticks(ticks=range(len(value_counts)), labels=value_counts.index.astype(str))
plt.savefig('Plot-PEGI.png')


print("Creating bar chart of Rating values")
plt.clf()
value_counts = df['Rating'].value_counts().sort_index()
plt.bar(range(len(value_counts)), value_counts.values, tick_label=value_counts.index)
plt.title('App Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.xticks(ticks=range(len(value_counts)), labels=[f'{x:.2f}' for x in value_counts.index])
plt.savefig('Plot-Ratings.png')


print("Creating bar chart of InAppPurchase values")
plt.clf()
value_counts = df['InAppPurchase'].value_counts().sort_index()
x_positions = np.array([0.25, 0.75])
plt.bar(x_positions, value_counts.values, width=0.2, tick_label=value_counts.index)
plt.title('In-App Purchases')
plt.xlabel('In App Purchase')
plt.ylabel('Frequency')
plt.xlim(0, 1)
plt.savefig('Plot-InAppPurchases.png')
```

```
47
48
49 print("Creating bar chart of Category values")
50 plt.clf()
51 value_counts = df['Category'].value_counts()
52 sorted_counts = value_counts.sort_values(ascending=False)
53 plt.bar(sorted_counts.index, sorted_counts.values)
54 plt.title('App Categories')
55 plt.xlabel('Category')
56 plt.ylabel('Frequency')
57 plt.savefig('Plot-Categories.png')
58
59
60 print("Creating horizontal bar chart of Genre values")
61 plt.clf()
62 all_values = df['Genres'].str.split(',').explode().str.strip()
63 value_counts = all_values.value_counts()
64 sorted_counts = value_counts.sort_values(ascending=True)
65 plt.barh(sorted_counts.index, sorted_counts.values)
66 plt.title('App Genres')
67 plt.tight_layout()
68 plt.savefig('Plot-Genres.png')
69
70
71
72 print("Creating chart of Sensor and Device values")
73 plt.clf()
74 dfSensor = df[['Microphone', 'Storage', 'Location', 'Bluetooth', '
       SpatialData', 'HandTracking', 'EyeTracking']]
75 y_counts = dfSensor.apply(lambda x: (x == 'Y').sum()).sort_values(
       ascending=True)
76
77 total_y = y_counts.sum()
78 percentage_y = (y_counts / total_y * 100).sort_values(ascending=True)
79
80 bars=plt.barh(percentage_y.index, percentage_y.values)
81
82 plt.title('Collection of Sensor and Device Data')
83 plt.xlabel('Percentage of apps collecting data')
84 plt.xticks(range(0, 41, 10))
85
86 # Annotate the bars with the percentage value
87 total = sorted_counts.sum()  # Calculate total for percentage
       conversion
88 for bar in bars:
89   percentage = bar.get_width() / total * 100
90   plt.text(bar.get_width(), bar.get_y() + bar.get_height()/2,
91            f'{percentage:.1f}%',
92            va='center', ha='left')
93
94 plt.tight_layout()
95 plt.savefig('Plot-SensorData.png')
96
97
98
99
100 print("Creating chart of User Data values")
101 plt.clf()
```

```
dfUserData = df[['UserID', 'UserProfile', 'Avatar', 'Followers', '
    UsageData', 'AgeGroup']]
y_counts = dfUserData.apply(lambda x: (x == 'Y').sum()).sort_values(
    ascending=True)

total_y = y_counts.sum()
percentage_y = (y_counts / total_y * 100).sort_values(ascending=True)

bars=plt.barh(percentage_y.index, percentage_y.values)
plt.title('Collection of User Data')
plt.xlabel('Percentage of apps collecting data')
plt.xticks(range(0, 41, 10))

# Annotate the bars with the percentage value
total = sorted_counts.sum()  # Calculate total for percentage
    conversion
for bar in bars:
  percentage = bar.get_width() / total * 100
  plt.text(bar.get_width(), bar.get_y() + bar.get_height()/2,
             f'{percentage:.1f}%',  # Formatting the number to one
    decimal place with a '%' symbol
             va='center', ha='left')

plt.tight_layout()
plt.savefig('Plot-UserData.png')


print("Creating bar chart of Qty Data values")
plt.clf()

frequency = df['Qty'].value_counts().sort_index()
# Ensure all values from 0 to 10 are included, even if some counts are
    zero
frequency = frequency.reindex(range(0, 11), fill_value=0)
plt.bar(frequency.index, frequency.values)
plt.title('Number of data items collected')
plt.xlabel('Data Items')
plt.ylabel('Frequency')
plt.xticks(range(0, 11))

plt.savefig('Plot-QtyData.png')

qtyCol=df["Qty"]
plt.figure(figsize=(10, 6))

# Plot histogram
qtyCol.hist(bins=range(11), density=True, alpha=0.6, color='
    midnightblue')

# Plot density
sns.kdeplot(qtyCol, color="black", lw=2, bw_adjust=0.5)

# Adding labels and title
plt.xlabel('Data Items')
plt.ylabel('Density')
plt.title('Number of data items collected')
plt.xticks(range(11))
plt.savefig('Plot-QtyData-Density.png')
```

```
154
155 print("Finished")
```

4.2 Plots of Privacy Policy data

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  from pandas.api.types import CategoricalDtype
5  import seaborn as sns
6  from scipy.stats import shapiro
7
8
9  df = pd.read_excel('MetaAppsData.xlsx', sheet_name='Analysis')
10
11
12 print("Creating chart of Compliance (Yes/No)")
13 plt.clf()
14 plt.figure(figsize=(6, 6))
15 value_counts = df['IsCompliant'].value_counts()
16 sorted_counts = value_counts.sort_values(ascending=False)
17 plt.bar(sorted_counts.index, sorted_counts.values)
18 plt.title('App Compliance')
19 plt.xlabel('Compliant?')
20 plt.ylim(0, 50)
21 plt.ylabel('Count')
22 plt.savefig('Plot-AppComplianceYesNo.png')
23
24
25 print("Creating chart of Compliance (Apps that do process data)")
26 plt.clf()
27 plt.figure(figsize=(6, 6))
28 df_processesData = df[df['ProcessesData'] == 'Yes']
29 value_counts = df_processesData['IsCompliant'].value_counts()
30 sorted_counts = value_counts.sort_values(ascending=False)
31 plt.bar(sorted_counts.index, sorted_counts.values)
32 plt.title('App Compliance')
33 plt.xlabel('Compliant?')
34 plt.ylim(0, 50)
35 plt.ylabel('Count')
36 plt.savefig('Plot-AppComplianceYesNoAndData.png')
37
38
39 print("Creating stacked bar chart of IsEU and IsCompliant")
40 plt.clf()
41 plt.figure(figsize=(6, 4))
42 # Prepare the data
43 pivot_data = df.pivot_table(index='IsEU', columns='IsCompliant',
       aggfunc='size', fill_value=0)
44
45 # Plot the data as a stacked bar chart
46 colors = ['red', 'lightgreen']
47 pivot_data.plot(kind='bar', stacked=True, color=colors, figsize=(10, 7)
       )
48 plt.title('Compliance Status by EU Membership')
49 plt.xlabel('Is EU Member')
```

```
plt.ylabel('Count')
plt.xticks(rotation=0)  # Rotates the x-axis labels to horizontal
plt.legend(title='Is Compliant')
plt.tight_layout()
plt.savefig('Plot-AppEUandCompliance.png')


print("Creating density plot of compliance")
plt.clf()
scoreCol=df["IntCompliance"]
plt.figure(figsize=(10, 6))

# Plot histogram
scoreCol.hist(bins=10, density=True, alpha=0.6, color='midnightblue')

# Plot density
sns.kdeplot(scoreCol, color="black", lw=2, bw_adjust=0.5)

# Adding labels and title
plt.xlabel('Compliance Level')
#plt.ylabel('Frequency')
plt.title('App Compliance')
plt.tight_layout()
plt.savefig('Plot-AppComplianceDensity.png')


print("Creating chart of Controller Locations")
plt.clf()
value_counts = df['Controller Location'].value_counts().sort_index()
sorted_counts = value_counts.sort_values(ascending=True)
plt.barh(sorted_counts.index, sorted_counts.values)
plt.title('Controller Locations')
plt.tight_layout()
plt.savefig('Plot-AppControllerLocations.png')


print("Creating stacked chart of popularity and Yes/No")
plt.clf()

bins = [1, 14, 27, 40, 53, 66]  # Define the bins
labels = ['1-13', '14-26', '27-39', '40-52', '53-65']  # Define labels
    for the bins
df['Popularity_Bin'] = pd.cut(df['Popularity'], bins=bins, labels=
    labels, right=False)

# Step 3: Group by the new 'Popularity_Bin' and count occurrences of '
    Yes' and 'No'
compliance_counts = df.groupby(['Popularity_Bin', 'IsCompliant']).size
    ().unstack(fill_value=0)

# Step 4: Create the stacked bar chart
ax = compliance_counts.plot(kind='bar', stacked=True, figsize=(10, 6),
    color=['#F44336','#4CAF50'])

# Step 4: Customize the chart
plt.title('Popularity vs Compliance')
plt.xlabel('Popularity')
plt.ylabel('Count')
```

```
plt.xticks(rotation=45)
plt.legend(title='Is Compliant', labels=['No', 'Yes'])
plt.grid(axis='y')

plt.tight_layout()
plt.savefig('Plot-AppPopularity.png')

print("Finished")
```

4.3 Analysis of Qty and Compliance values

```
import numpy as np
import pandas as pd
import scipy.stats as stats
from scipy.stats import shapiro



def columnStats(myCol, myColName):
    print("========================================\n")
    print("Column stats for ", myColName)
    print("Mean :", myCol.mean())
    print("Median :", myCol.median())
    print("Mode :", myCol.mode())
    print("Variance : ", myCol.var())
    print("Std dev :", myCol.std())
    print("Skew :", myCol.skew())


    # Shapiro-Wilk test
    w_statistic, p_value = shapiro(myCol)
    print("W statistic:", w_statistic)
    print("P-value:", p_value)


    # Calculate the 95% Confidence Interval for the mean
    sample_mean = myCol.mean()
    sample_std = myCol.std()
    n = len(myCol)

    confidence_level = 0.95
    z_critical = stats.norm.ppf((1 + confidence_level) / 2)  # Two-
   tailed z-value
    margin_error = z_critical * (sample_std / np.sqrt(n))

    ci_lower = sample_mean - margin_error
    ci_upper = sample_mean + margin_error

    print("95% Confidence Interval:", ci_lower, ci_upper)

    print("========================================\n")


df = pd.read_excel('MetaAppsData.xlsx', sheet_name='Selected', skiprows
   =[0])
qtyCol = df["Qty"]
```

```
44 columnStats(qtyCol, "Qty")
45
46 df = pd.read_excel('MetaAppsData.xlsx', sheet_name='Analysis')
47 complianceCol = df["IntCompliance"]
48 columnStats(complianceCol, "Compliance")
```

References

[1] Overleaf, "Overleaf, Online LaTeX editor." [Online]. Available: https://www.overleaf.com/

[2] VMware, "VMWare Workstation Pro." [Online]. Available: https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion

[3] Debian, "Debian Linux." [Online]. Available: https://www.debian.org/

[4] Sublime, "Sublime Text Editorditor." [Online]. Available: https://www.sublimetext.com/

[5] Python Software Foundation, "Python." [Online]. Available: https://www.python.org

[6] Posit, "RStudio." [Online]. Available: https://posit.co/download/rstudio-desktop/

[7] The R Foundation, "R Language." [Online]. Available: https://www.r-project.org/

[8] Microsoft, "Microsoft Office." [Online]. Available: https://www.microsoft365.com

[9] Meta, "Shop Meta Quest VR games, apps, deals and more." [Online]. Available: https://www.meta.com/en-gb/experiences/