Configuration Manual

MSc Research Project

MSc FinTech

Sunday Adetunji

Student ID: X21232571

School of Computing

National College of Ireland

Supervisor: Brian Byrne

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sunday Adeyinka Adetunji
Student ID:	X21232571
Programme:	MSc. FinTech **Year:** 2023.
Module:	MSc. Research Project
Lecturer:	Brian Byrne
Submission Due Date:	14/08/2023
Project Title:	An Investigation on the Impact of the Revised Payment Services Directive (PSD2) on Payment Services in Ireland
Word Count:	7972 **Page Count:** 26

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: Sunday Adetunji

Date: 13/08/2023

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

Sunday Adetunji
Student ID: X21232571

1 Introduction

A user configuration manual that encompasses intricate technical details, specifications, and step-by-step processes essential for reproducing the research analysis titled " An Investigation on the Impact of the Revised Payment Services Directive (PSD2) on Payment Services in Ireland".

2 System Requirement

2.1 Hardware Specification

The hardware requirement for this study is that of Windows 10 Operating System (OS) and these are 2 GB of RAM, 1 GHz or faster processor and 20 GB HDD or SDD.

The hardware configuration of the machine used to implement this study is given as

- Processor – 11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz 2.42 GHz
- Memory – 20.0 GB
- Storage - 1 TB HDD

2.2 Software Specification

The software requirements for this project include Operation System (OS), Document Processor, Programming Language, and Web Browser Application. Below are the software packages used in the implementation of software packages.

- **Microsoft Windows 11 OS**
- **Python Programming Language –** version 3.10.12
- **Google Colab** – is a cloud-based platform that offers an interactive Jupyter Notebook environment for writing and executing code. It's designed to make it easy for researchers, developers, and students to collaborate on coding projects, particularly those involving machine learning and data analysis.
- **Mozilla Firefox Web Browser**
- **Microsoft Word**

3 Data

For the study, two sets of questionnaires were developed and used to collect primary from stakeholders in the financial sectors in Ireland and Consumers of payment services.

The stakeholders' questionnaire aims to analyze how PSD2 implementation impacts Irish payment services, including changes in customer behaviour and the adoption of new payment methods. PSD2 is a directive governing EU payment services, aimed at enhancing competition, consumer protection, and innovation. The questionnaires have 29 questions which the respondent is expected to complete.

The consumers' questionnaire aims to assess the impact of PSD2 implementation on payment services like cards, mobile apps, online banking, and crypto payments in Ireland, including shifts in consumer behaviour and new payment method adoption. PSD2, introduced in January 2018, regulates EU payment activities to boost competition, consumer protection, and innovation. The questionnaires have 24 questions which the respondent is expected to complete

4 Analysis

4.1 Python's Libraries/Packages Required for Analysis

Library Name	Version	Use
google-colab	1.0.0	Google Collaboratory library contains several other libraries such as auth used to request authentication to access Google Drive to access Google work sheets.
google-auth	2.17.3	Google authentication library was used to access authentication credentials to facilitate gspread authorization.
gspread	3.4.2	GSpread is a Python API from Google to access and manipulate Google Sheets. The worksheet was retrieved via this library

seaborn	0.12.2	Seaborn, built upon matplotlib, offers a user-friendly interface for creating visually appealing and informative statistical graphics in Python.
pandas	1.5.3	Pandas' library serves as a vital tool for data manipulation, analysis, preprocessing, and dataset preparation before using statistical analysis.
numpy	1.23.5	NumPy is used for numerical computations providing efficient array operations and mathematical functions that enhance performance and ease of scientific computing.
matplotlib	3.7.1	This library was used alongside Seaborn to create descriptive statistic visualizations such as Pie charts and Bar charts.
statsmodels	0.14.0	Statsmodels is used for statistical modelling and hypothesis testing in Python, offering a comprehensive toolkit for analysing data and deriving meaningful insights from various statistical techniques. The ANOVA and ordinary least squares (OLS) regression were imported for this module.
Ipython	7.34.0	This library contains the HTML module used to centre the plots or chart diagram generated.

Table 1: Import Python Libraries/Packages

4.2 Defined Global Variables and Functions

4.2.1 Global Variables

To preprocess the questionnaire responses for the different question formats, variables ***question_5_alternate***, ***stakeholders_responses_map*** and ***psd2_investigation_ responses_map*** were created. ***question_5_alternate*** is a Python list used to hold stakeholders' question 5 variants. ***stakeholders_responses_map*** and ***psd2_investigation_ responses_map*** variables were used to storekey–value dictionary objects for encoding the options in each question for both questionnaires.

4.2.2 Functions

- ***draw_line*** function – This function is used to draw a horizontal bar or line of a given size. The size is determined by the magnitude of the parameter given to the function.

- ***get_all_colors*** function – This is used to return a list of colours used to differentiate the colour of each component in a bar chart.

- ***generate_pmap_values*** function – This function is used to generate a key-value pair dictionary data used to encode the category or level for a given question. It accepts one parameter which is a list of the categories in the question and encodes them with integer values starting from one. The created dictionary data is used to toggle the levels to the original categories or the integer equivalent.

- ***rearrange_questions_column_names*** function – This function is used to group question 5 alternative one question and use a dictionary to map the question to a unique column name. Question 5 contains a list of 5 unique values, one for each alternate.

4.3 Data Pre-processing and Transformation

4.3.1 Load Datasets from Google Drive

To begin the preprocessing phase, the data must first be loaded into our workspace. The code snippet below shows how the questionnaire data were loaded into our workspace.

Load Google Form Responses into Google Colab

```
[83] wb_responses_from_stackholders = gc.open_by_key('18-abSXc7auiheAU-8HmkgmyIqlW_a3KToR-LhNX8CDI')
     wb_responses_from_investigation_on_impact_of_psd2 = gc.open_by_key('1_yz291QddD8-8MWi-mofmFvtqIlCBPK6DbEhIz5EGzQ')

     ws_responses_from_stackholders = wb_responses_from_stackholders.worksheet("stakeholders")
     ws_responses_from_investigation_on_impact_of_psd2 = wb_responses_from_investigation_on_impact_of_psd2.worksheet("rpsireland")

     responses_from_stackholders_rows = ws_responses_from_stackholders.get_all_values()
     investigation_on_impact_of_psd2_rows = ws_responses_from_investigation_on_impact_of_psd2.get_all_values()
```

Figure 1: Loading datasets from google sheets

Figure 1 above, shows the use of gspread library to load the responses to the questionnaires into Google Colab Notebook. The instance of gspread module ***gc*** was created and ***open_by_key*** method was invoked to load the Google sheet document from Google Drive. To retrieve the data record in the target worksheet, the ***worksheet*** method and ***get_all_values*** method were called on the documents that have been read as shown in figure 1.

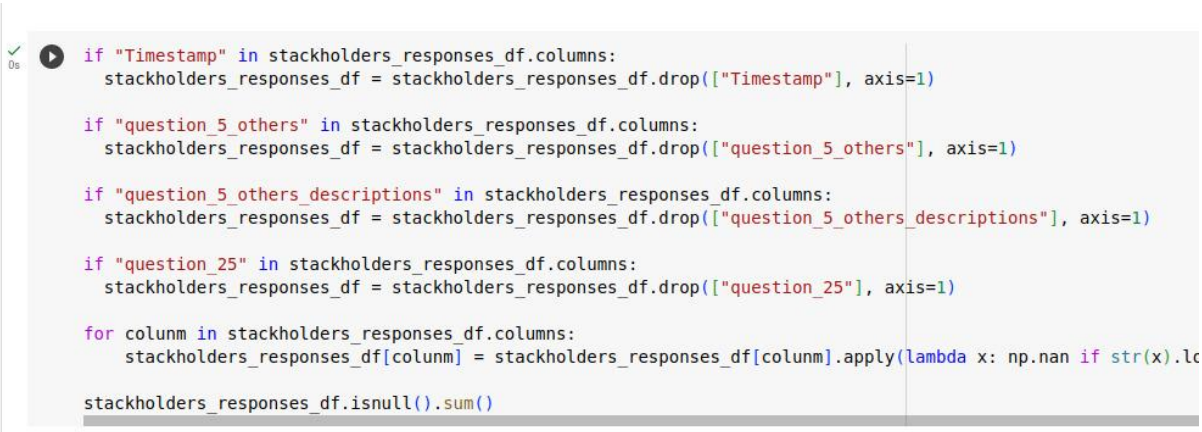
```
#
# Stakeholders Responses Preprocessing
#
questions_responses_from_stackholders = responses_from_stackholders_rows[0]

rs_questions_to_columns_dict = dict()

for ind in range(len(questions_responses_from_stackholders)):
  label = "question_{}".format(ind)
  if ind == 0:
    label = questions_responses_from_stackholders[ind]
  elif (ind >= 5) and (ind <= 10):
    alternate_ind = ind - 5
    label = "question_5_{}".format(question_5_alternate[alternate_ind])
  elif ind > 10:
    label = "question_{}".format(ind-5)

  rs_questions_to_columns_dict[label] = questions_responses_from_stackholders[ind]

stackholders_responses_df = pd.DataFrame.from_records(responses_from_stackholders_rows[1:], columns=list(rs_questions_to_columns_dict.keys()))


#
# Customers Responses Preprocessing
#
questions_investigation_on_impact_of_psd2 = investigation_on_impact_of_psd2_rows[0]

psd2_questions_to_columns_dict = dict()

for ind in range(len(questions_investigation_on_impact_of_psd2)):
  label = "question_{}".format(ind)
  if ind == 0:
    label = questions_investigation_on_impact_of_psd2[ind]
  psd2_questions_to_columns_dict[label] = questions_investigation_on_impact_of_psd2[ind]

investigation_psd2_df = pd.DataFrame.from_records(investigation_on_impact_of_psd2_rows[1:], columns=list(psd2_questions_to_columns_dict.keys()))
```

Figure 2: Process load data from google sheets into Pandas Dataframe

The data loaded from Google Sheets are read as a list. Figure 2 above shows the transformation of these lists to a pandas data frame with the first record in the list set as the pandas' data frame column value. The first step here was to retrieve the first records in the data list from the read Google sheet which contains the columns data. The column for both datasets were then used to create ***rs_questions_to_columns_dict*** for stakeholders' data and ***psd2_questions_to_columns_dict*** for consumers' data dictionary data whose keys are ***question_1***, ***question_2***, etc. and the values are the questions themselves. The analysis data frames ***stakeholders_responses_df*** and ***investigation_psd2_df*** were created using ***pd.Dataframe.from_records*** using the appropriate data records.

4.3.2 Remove Columns

After loading the datasets as pandas' dataframe, the ***Timestamp***, ***question_5_others***, ***question_5_others_description*** and ***question_25*** were dropped from the ***stakeholders_responses_df*** dataset. From the ***investigation_psd2_df, the Timestamp*** was dropped as well. Timestamp was removed because they contain timestamp data and are not relevant to the analysis, the other column deleted was because they were sparsely populated and may impact our analysis negatively.



```
if "Timestamp" in stackholders_responses_df.columns:
  stackholders_responses_df = stackholders_responses_df.drop(["Timestamp"], axis=1)

if "question_5_others" in stackholders_responses_df.columns:
  stackholders_responses_df = stackholders_responses_df.drop(["question_5_others"], axis=1)

if "question_5_others_descriptions" in stackholders_responses_df.columns:
  stackholders_responses_df = stackholders_responses_df.drop(["question_5_others_descriptions"], axis=1)

if "question_25" in stackholders_responses_df.columns:
  stackholders_responses_df = stackholders_responses_df.drop(["question_25"], axis=1)

for columm in stackholders_responses_df.columns:
    stackholders_responses_df[columm] = stackholders_responses_df[columm].apply(lambda x: np.nan if str(x).lc

stackholders_responses_df.isnull().sum()
```

Figure 3: Code snippet showing the removal of columns from stakeholders' data frame

```
if "Timestamp" in investigation_psd2_df.columns:
  investigation_psd2_df = investigation_psd2_df.drop(["Timestamp"], axis=1)

for columm in investigation_psd2_df.columns:
    investigation_psd2_df[columm] = investigation_psd2_df[columm].apply(lambda x: np.nan if str(x).lower()==""

investigation_psd2_df.isnull().sum()
```

Figure 4: Code snippet showing the removal of columns from customers' data frame

4.3.3 Handle Missing Data

Missing data were in the form of **nan**, empty string, and badly formatted input. All empty strings and badly formatted data were replaced with ***np.nan*** data. The two datasets were checked for the occurrence of missing data before attempting to replace all missing data with the mode value for any given pandas' series.

```
for columm in stackholders_responses_df.columns:
  stackholders_responses_df[columm] = stackholders_responses_df[columm].fillna(stackholders_responses_df[columm].mode().iloc[0])

stackholders_responses_df.isnull().sum()
```

Figure 5: Handling missing by replacing missing data with mode value for stakeholders' data frame

```
for columm in investigation_psd2_df.columns:
  investigation_psd2_df[columm] = investigation_psd2_df[columm].fillna(investigation_psd2_df[columm].mode().iloc[0])

stackholders_responses_df.isnull().sum()
```

Figure 6: *Handling missing by replacing missing data with mode value for customers' data frame*

4.3.4 Transform Pandas' Object Datatype to Pandas' Integer Datatype

To transform non-numerical data (string) to numerical data, pandas' data frame ***map()*** method transforms values within a specified column or series in the data frame. The map function accepts dictionary data where all unique values to be replaced are the keys of the dictionary and the replace values are the assigned values. The ***generate_pmap_values()*** was used to generate this dictionary object. This operation is done for all object datatypes in both datasets.

```
object_columns = investigation_psd2_df.select_dtypes("object").columns

for column in object_columns:
  investigation_psd2_df[column] = investigation_psd2_df[column].str.lower()
  unique_values = investigation_psd2_df[column].unique()
  map_values = generate_pmap_values(unique_values)
  psd2_investigation_responses_map[column] = map_values
  investigation_psd2_df[column] = investigation_psd2_df[column].map(map_values)
```

```
[158] investigation_psd2_df = investigation_psd2_df.astype('int',copy=False)

      investigation_psd2_df.info()
```

Figure 7: Data transformation for object datatype to integer datatype for stakeholders'dataframe

```
[153] object_columns = stackholders_responses_df.select_dtypes("object").columns

      for column in object_columns:
        if "question_5_" in column:
          pass
        else:
          stackholders_responses_df[column] = stackholders_responses_df[column].str.lower()
          unique_values = stackholders_responses_df[column].unique()
          map_values = generate_pmap_values(unique_values)
          stakeholders_responses_map[column] = map_values
          stackholders_responses_df[column] = stackholders_responses_df[column].map(map_values)
```

```
[154] stackholders_responses_df = stackholders_responses_df.astype('int',copy=False)

      stackholders_responses_df.info()
```

Figure 8: *Data transformation for object datatype to integer datatype for customers'dataframe*

```
[146] stackholders_responses_df.head()
```

	Timestamp	question_1	question_2	question_3	question_4	question_5_security_of_customers	question_5_innovation	question_5_competition	question_5_profitabilit
0	5/2/2023 17:55:43	Yes	Necessary	Innovation	Positive	5	4	2	
1	5/3/2023 6:33:16	Yes	Necessary	Innovation	Positive	3	5	5	
2	5/3/2023 8:19:43	Yes	Necessary	Security of consumers	Positive	4	2	2	
3	5/3/2023 8:30:56	Yes	Necessary	Security of consumers	Positive	3	1	2	
4	5/3/2023 9:16:53	Yes	Necessary	Security of consumers	Positive	5	2	1	

0s completed at 7:21 PM

Figure 9: Stakeholders' data frame before transformation

```
[149] investigation_psd2_df.head()
```

	Timestamp	question_1	question_2	question_3	question_4	question_5	question_6	question_7	question_8	question_9	...	question_15	question_16	question_17
0	6/24/2023 7:42:51	Yes	Yes	After 2018	Cards	Speed of transaction	Payments	Yes		Yes	...	Not applicable.	I am not sure.	Not sure.
1	6/25/2023 15:28:56	No	Yes	Before 2018	Mobile apps	Convenience	Payments	Yes	No	Yes	...	I am not sure	I am not sure.	Not sure.
2	6/25/2023 15:36:05	Yes	Yes	After 2018	Cards	Convenience	Payments	Yes	No	Yes	...	I feel more secure now.	Security of payments now is better than how it...	Payment service is more convenient since 2018.
3	6/25/2023 15:40:39	No	Yes	After 2018	Cards	Speed of transaction	Payments	Yes	No	Yes	...	I am not sure	Security of payments now is better than how it...	Payment service is more convenient since 2018.
4	6/25/2023 15:44:28	No	Yes	After 2018	Cards	Convenience	Payments	No	No	No	...	I am not sure	I am not sure.	Not sure.

5 rows × 25 columns

Figure 10: Customers' data frame before transformation

```
stackholders_responses_df.head()
```

	question_1	question_2	question_3	question_4	question_5_security_of_customers	question_5_innovation	question_5_competition	question_5_profitability	question
0	1	1	1	1	5	4	2	2	
1	1	1	1	1	3	5	5	5	
2	1	1	2	1	4	2	2	3	
3	1	1	2	1	3	1	2	1	
4	1	1	2	1	5	2	1	2	

5 rows × 31 columns

```
[160] stackholders_responses_df.shape
```

```
(53, 31)
```

```
[161] investigation_psd2_df.head()
```

	question_1	question_2	question_3	question_4	question_5	question_6	question_7	question_8	question_9	question_10	...	question_15	question_16	question_17
0	1	1	1	1	1	1	1	1	1	1	...	1	1	1
1	2	1	2	2	2	1	1	1	1	1	...	2	1	1
2	1	1	1	1	2	1	1	1	1	2	...	3	2	2
3	2	1	1	1	1	1	1	1	1	1	...	2	2	2
4	2	1	1	1	2	1	2	1	2	1	...	2	1	1

5 rows × 24 columns

Figure 11: Stakeholders' and Customers' data frames after transformation

4.4 Descriptive Statistics

Descriptive statistics was the distribution frequency for the selected categories within a given question. The ***matplotlib.pyplot*** was used to plot three (3) different type of charts and these includes Pie Charts, Bar Charts and Grouped Bar Charts. The Pie charts were used to show the distribution frequencies in percentages for the categories in a given question. The Bar charts were used to show the distribution count for the categories in a given question. The Grouped Bar chart was used to show the distribution count for the categories in a given sub-questions in question 5 in the stakeholders' dataset. The figure below shows some of the descriptive statistic charts.

```
[ ]  for column in investigation_psd2_df:

       fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(25, 8))
       axes = axes.ravel()

       res_opt_size = investigation_psd2_df[column].nunique()
       question_response_dist = list()
       labels = list()
       colors = list()
       key_count = -1
       value_counts = investigation_psd2_df[column].value_counts()
       question_map_swap = {v: k for k, v in psd2_investigation_responses_map[column].items()}

       for key in value_counts.keys():
         if column in psd2_excludes:
           label = "invalid" if int(key) < 0 else str(key).upper()
         else:
           label = "invalid" if int(key) < 0 else question_map_swap[key]
         labels.append(label.upper())
         key_count += 1
         colors.append(get_all_colors()[key_count])
         question_response_dist.append(value_counts[key])

       print("\n\n")
       question_title = " ".join(column.split("_")).upper()
       print(question_title)
       print(draw_line(len(question_title)))
       print("\n")
       print(psd2_questions_to_columns_dict[column].upper())
       print("\n\n")

       for index, ax in enumerate(axes):
         if index == 0:
           ax.pie(question_response_dist, labels=labels, autopct="%1.1f%%")
           ax.legend(loc=0)
         elif index == 1:
           ax.bar(labels, question_response_dist, color=colors, label=labels)
           ax.set_ylabel("FREQUENCY")
           ax.tick_params(axis='x', colors='w')
           ax.legend(loc=0)

       fig.tight_layout()
       plt.show()
       print("\n\n\n\n\n")
```

Figure 12: Code snippet for showing Pie chart and Bar charts for the descriptive statistics

4.5 Hypothesis Testing

Four different hypothesis testing was carried out using ordinary least square (OLS) regression and ANOVA (Analysis of Variance) to measure and determine if the is a linear relationship between measurement variables ("security of customers", "competitiveness", "innovation" and "profitability" as the independent variables) and the rest of variables in the datasets as the independent variables. The linear regression model was built from the ***ols*** module from ***statsmodels.formula.api*** library. The ***ols*** function accepts two parameters ***formula*** and ***data*** which create ***mlr*** variable to store the regression model instance**.** The formula parameter specifies the dependent variable and independent variables. The ***mlr*** model fit method

generates the model summary data. To perform the ANOVA test, a new regression model only has an intercept or a constant. The new regression model is fitted and stored in the variable ***anova_lr***. The ANOVA model is created using ***statsmodels.api.stats.anova_lm*** module which accepts 4 paramenters ***anova_lr, mlr, test*** parameter is set to "F", ***typ*** parameter is set to "I".

```
[ ]  # Retrieve the dependent Variable
     dependent_variable = stackholders_responses_df[[stackholders_responses_df.columns[4
     # Retrieve the independent Variables
     independent_variables = stackholders_responses_df.drop(stackholders_responses_df.co

     # define formula string
     formula_str = "{} ~ ".format(dependent_variable.columns[0])
     for index in range(len(independent_variables.columns)):
       if index == 0:
         formula_str += "{} ".format(independent_variables.columns[index])
       else:
         formula_str += "+ {} ".format(independent_variables.columns[index])

     # Perform linear regression using ordinary least squares (OLS) regressor
     mlr = smf.ols(formula=formula_str, data=stackholders_responses_df).fit()

     # Perform the analysis of variance
     formula_str = "{} ~ 1".format(dependent_variable.columns[0])
     anova_lr = smf.ols(formula=formula_str, data=stackholders_responses_df).fit()
     anova_lr = sm.stats.anova_lm(anova_lr, mlr, test="F", typ="I")

     print()
     print("ANOVA Result Summary")
     print("====================")
     print()
     print(anova_lr)
     print()
     print()
     print()
     print()
     print()
     print(mlr.summary())
     print()
     print()
```

Figure 13: Code Snippet for hypothesis testing

5 Conclusion

The instructions provided in this configuration manual documentation will assist any researcher aiming to replicate the code implementation outlined in this research paper. By utilizing the provided datasets, one can expect to achieve identical results to those obtained in this study. The code examples and concise table presented in the paper were instrumental in fulfilling the project's established objectives and aims.