National College of Ireland

Configuration Manual

MSc Research Project

MSc in Cyber Security

Trecy Soumya Charles

Student ID: x21143650

School of Computing

National College of Ireland

Supervisor: Mr. Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Trecy Soumya Charles
Student ID:	X21143650
Programme:	MSc in Cyber Security **Year:** 2022-23
Module:	MSc Research Project
Supervisor:	Mr. Michael Pantridge
Submission Due Date:	15.12.2022
Project Title:	Proficient User Authentication based on Dynamic keystroke using Machine Learning

Word Count:492 **Page Count :9**

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: Trecy Soumya Charles

Date: 15.12.2022

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

Trecy Soumya Charles
Student ID: x21143650

1 Introduction

The setup manual is crucial for providing details on all the gear, software, and processes required to carry out this project. The research project uses machine learning methods to study the dynamics of keystrokes and heavily consults the setup guide. This manual contains the standard setup and requires equipment for performing this operation.

2. Hardware Requirements

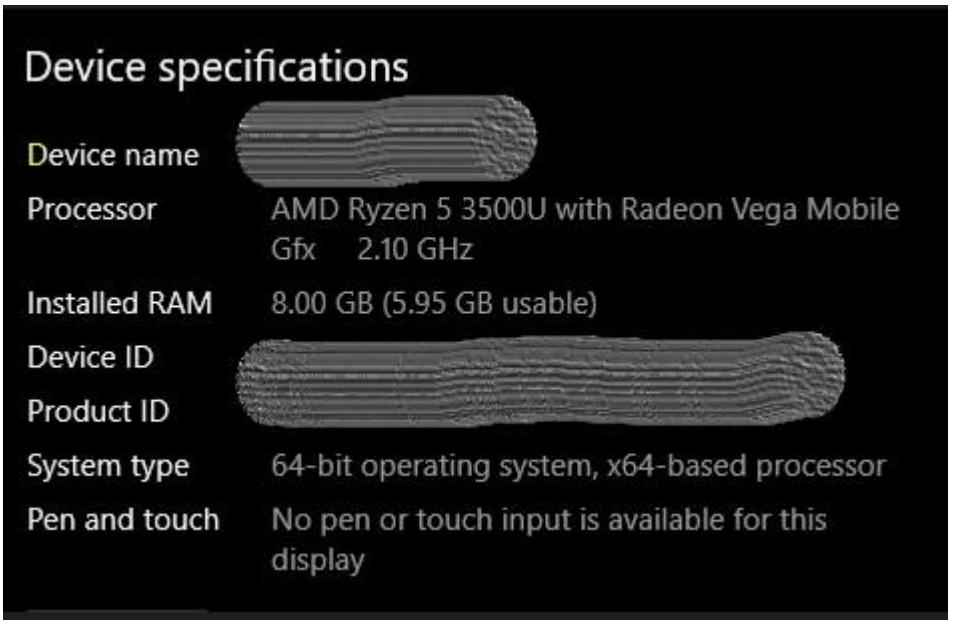

Fig 1: Hardware Details

3. Software Requirements

Fig 2: Software details

Any operating system that supports the Python IDE (Google Collaboratory) is advised. One of the simpler operating systems to comprehend and set up is Windows.

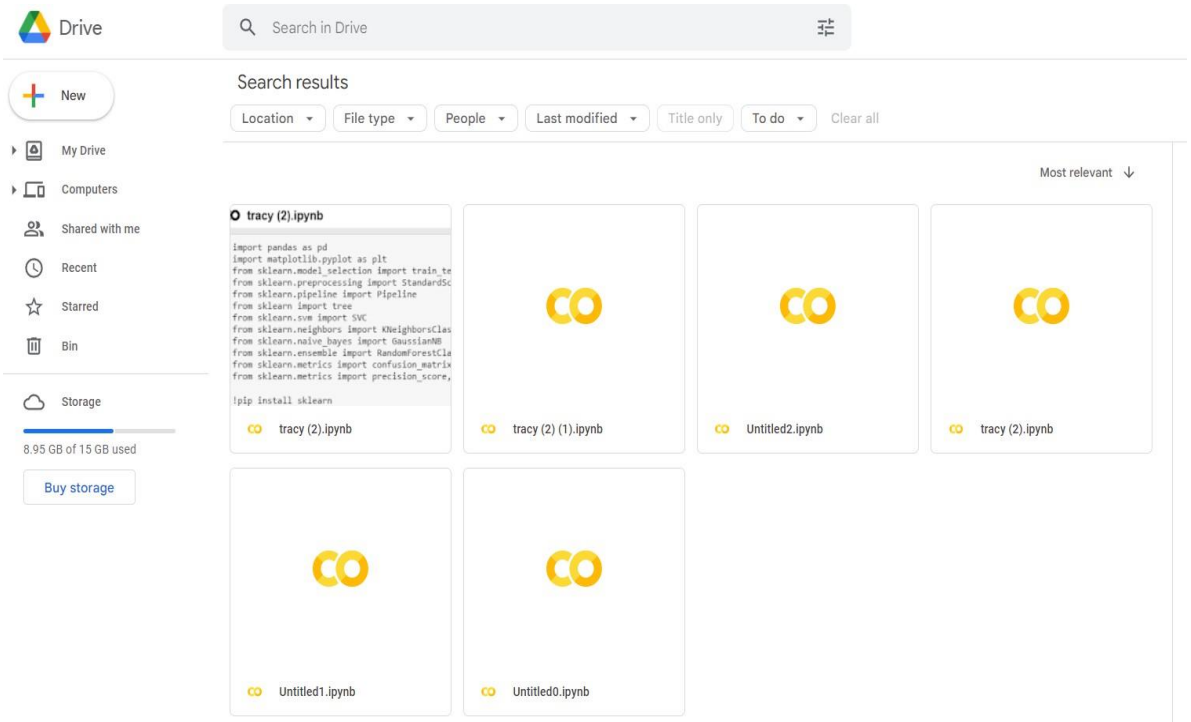

Fig 3: Google Collab UI

4. Library Packages

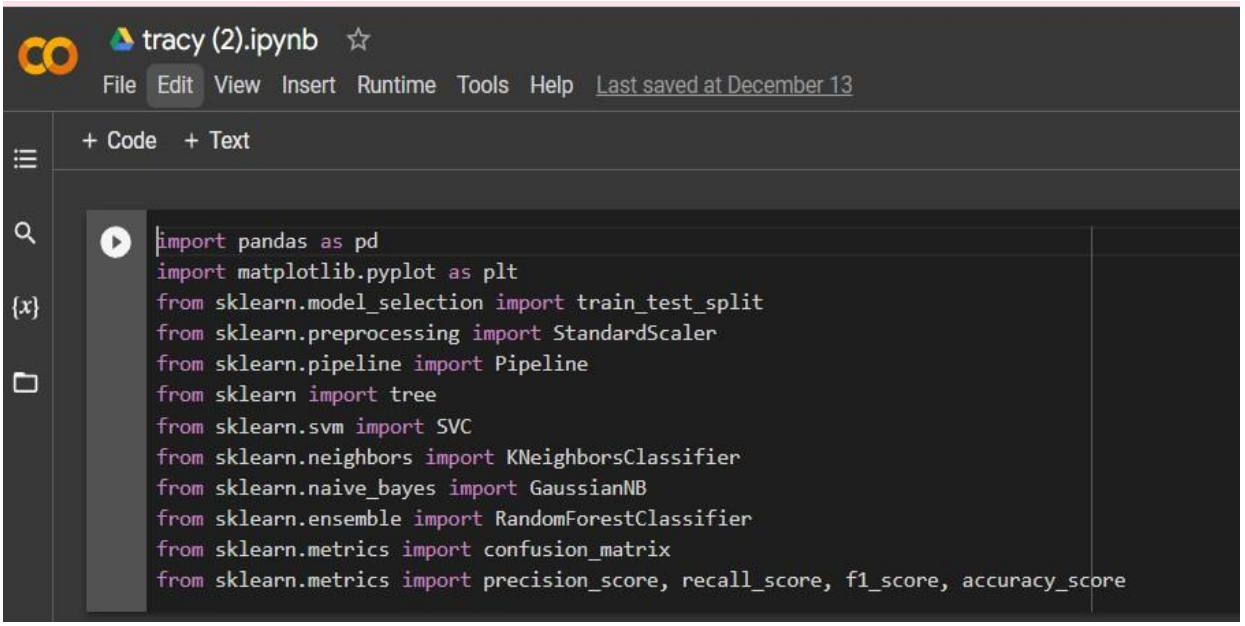

Fig 4: Importing Libraries

Above are the libraries imported to employ the functions available in it to build the implementation.

```
[3] from google.colab import files
    uploaded = files.upload()
```

Choose Files DSL-Strong...ordData.csv

- **DSL-StrongPasswordData.csv**(text/csv) - 4669935 bytes, last modified: 12/10/2022 - 100% done

Saving DSL-StrongPasswordData.csv to DSL-StrongPasswordData.csv

Fig 5: uploading the dataset

upload the dataset which is in the .CSV file for the evaluation.

```
df=pd.read_csv('DSL-StrongPasswordData.csv')
```

+ Code + Text

```
df.head()
```

	subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	H.t	DD.t.i	UD.t.i	H.i	...	H.a	DD.a.n	UD.a.n	H.n	DD.n.l	UD.n.l	H.l	DD.l.Return	UD.l.Return	H.Return
0	s002	1	1	0.1491	0.3979	0.2488	0.1069	0.1674	0.0605	0.1169	...	0.1349	0.1484	0.0135	0.0932	0.3515	0.2583	0.1338	0.3509	0.2171	0.0742
1	s002	1	2	0.1111	0.3451	0.2340	0.0694	0.1283	0.0589	0.0908	...	0.1412	0.2558	0.1146	0.1146	0.2642	0.1496	0.0839	0.2756	0.1917	0.0747
2	s002	1	3	0.1328	0.2072	0.0744	0.0731	0.1291	0.0560	0.0821	...	0.1621	0.2332	0.0711	0.1172	0.2705	0.1533	0.1085	0.2847	0.1762	0.0945
3	s002	1	4	0.1291	0.2515	0.1224	0.1059	0.2495	0.1436	0.1040	...	0.1457	0.1629	0.0172	0.0866	0.2341	0.1475	0.0845	0.3232	0.2387	0.0813
4	s002	1	5	0.1249	0.2317	0.1068	0.0895	0.1676	0.0781	0.0903	...	0.1312	0.1582	0.0270	0.0884	0.2517	0.1633	0.0903	0.2517	0.1614	0.0818

5 rows × 34 columns

```
[ ] df.tail()
```

	subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	H.t	DD.t.i	UD.t.i	H.i	...	H.a	DD.a.n	UD.a.n	H.n	DD.n.l	UD.n.l	H.l	DD.l.Return	UD.l.Return	H.Return
20395	s057	8	46	0.0884	0.0685	-0.0199	0.1095	0.1290	0.0195	0.0945	...	0.1219	0.1383	0.0164	0.0820	0.1329	0.0509	0.1005	0.2054	0.1049	0.1047
20396	s057	8	47	0.0655	0.0630	-0.0025	0.0910	0.1148	0.0238	0.0916	...	0.1008	0.0512	-0.0496	0.1037	0.0868	-0.0169	0.1445	0.2206	0.0761	0.1198
20397	s057	8	48	0.0939	0.1189	0.0250	0.1008	0.1122	0.0114	0.0721	...	0.0913	0.1169	0.0256	0.0689	0.1311	0.0622	0.1034	0.2017	0.0983	0.0905
20398	s057	8	49	0.0923	0.1294	0.0371	0.0913	0.0990	0.0077	0.0992	...	0.0882	0.0821	-0.0061	0.0576	0.0697	0.0121	0.0979	0.1917	0.0938	0.0931
20399	s057	8	50	0.0596	0.1310	0.0714	0.0992	0.1103	0.0111	0.0998	...	0.0969	0.0784	-0.0185	0.0790	0.1133	0.0343	0.0807	0.1993	0.1186	0.1018

5 rows × 34 columns

Fig 6: Importing dataset

```
df.isnull()
```

	subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	H.t	DD.t.i	UD.t.i	H.i	...	H.a	DD.a.n	UD.a.n	H.n	DD.n.l	UD.n.l	H.l	DD.l.Return	UD.l.Return	H.Return
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20395	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
20396	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
20397	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
20398	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
20399	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

20400 rows × 34 columns

Fig 7: Checking is there any null value

```
[11] print(df['subject'].unique())

     ['s002' 's003' 's004' 's005' 's007' 's008' 's010' 's011' 's012' 's013'
      's015' 's016' 's017' 's018' 's019' 's020' 's021' 's022' 's024' 's025'
      's026' 's027' 's028' 's029' 's030' 's031' 's032' 's033' 's034' 's035'
      's036' 's037' 's038' 's039' 's040' 's041' 's042' 's043' 's044' 's046'
      's047' 's048' 's049' 's050' 's051' 's052' 's053' 's054' 's055' 's056'
      's057']
```

Fig 8: Unique values in Dataset

```
[17] import time
     start = time.time()

     dt_model = tree.DecisionTreeClassifier()
     dt_model.fit(X_train, y_train)

     end = time.time()
     time = end - start
     Final_Time = time/60
     print("time taken:", Final_Time, "minutes")
     print("time taken:", end - start)

     time taken: 0.015067009131113689 minutes
     time taken: 0.9040205478668213

[18] dt_model_pred=dt_model.predict(X_test)
```

Fig 9: decision tree model training

```
[19] print("Precision Score : ", precision_score(y_test, dt_model_pred,pos_label='positive', average='micro'))
     print("Recall Score : ", recall_score(y_test, dt_model_pred,pos_label='positive', average='micro'))
     print("F1 Score : ", f1_score(y_test, dt_model_pred,pos_label='positive', average='micro'))
     print("Accuracy Score : ", accuracy_score(y_test, dt_model_pred))

     Precision Score :  0.708407605466429
     Recall Score :  0.708407605466429
     F1 Score :  0.708407605466429
     Accuracy Score :  0.708407605466429
```

Fig 10: Model accuracy summary

```
svc_model = SVC( kernel='linear', gamma= 1)
svc_model.fit(X_train, y_train)
```

```
SVC(gamma=1, kernel='linear')
```

```
[ ]
```

```
svc_model_pred=svc_model.predict(X_test)
```

Fig 11: SVC model training

```
[ ] print("Precision Score : ", precision_score(y_test, svc_model_pred,pos_label='positive', average='micro'))
    print("Recall Score : ", recall_score(y_test, svc_model_pred,pos_label='positive', average='micro'))
    print("F1 Score : ", f1_score(y_test, svc_model_pred,pos_label='positive', average='micro'))
    print("Accuracy Score : ", accuracy_score(y_test, svc_model_pred))
```

```
Precision Score :  0.8609625668449198
Recall Score :  0.8609625668449198
F1 Score :  0.8609625668449198
Accuracy Score :  0.8609625668449198
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1370: UserWarning: Note that pos_l
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1370: UserWarning: Note that pos_l
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1370: UserWarning: Note that pos_l
  warnings.warn(
```

Fig 12: SVC Model accuracy summary

```
[ ] knn_model = KNeighborsClassifier(n_neighbors=3)
```

```
[ ] knn_model.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
[ ] knn_model_pred=knn_model.predict(X_test)
```

Fig 13: KNeighbors model training

```
[ ]  print("Precision Score : ", precision_score(y_test, knn_model_pred,pos_label='positive', average='micro'))
     print("Recall Score : ", recall_score(y_test, knn_model_pred,pos_label='positive', average='micro'))
     print("F1 Score : ", f1_score(y_test, knn_model_pred,pos_label='positive', average='micro'))
     print("Accuracy Score : ", accuracy_score(y_test, knn_model_pred))

     Precision Score :  0.8481877599524659
     Recall Score :  0.8481877599524659
     F1 Score :  0.8481877599524659
     Accuracy Score :  0.8481877599524659
     /usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1370: UserWarning: Note that pos_la
       warnings.warn(
     /usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1370: UserWarning: Note that pos_la
       warnings.warn(
     /usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1370: UserWarning: Note that pos_la
       warnings.warn(
```

Fig 14 Model accuracy summary

```
[ ]  gnb_model = GaussianNB()

[ ]  gnb_model.fit(X_train, y_train)

     GaussianNB()

[ ]  gnb_model_pred=gnb_model.predict(X_test)
```

Fig 15: Gaussian Model training

```
Precision Score :  0.6685977421271538
Recall Score :  0.6685977421271538
F1 Score :  0.6685977421271538
Accuracy Score :  0.6685977421271538
/usr/local/lib/python3.8/dist-packages/sklearn/m
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/m
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/m
  warnings.warn(
```

Fig 16: Model Accuracy summary

```
[ ]  rf_model = RandomForestClassifier(max_depth=15, random_state=0)

[ ]  rf_model.fit(X_train, y_train)

     RandomForestClassifier(max_depth=15, random_state=0)

[ ]  rf_model_pred=rf_model.predict(X_test)
```

Fig 17: Random Forest Classifier

```
print("Precision Score : ", precision_score(y_test, rf_model_pred,pos_label='positive', average='micro'))
print("Recall Score : ", recall_score(y_test, rf_model_pred,pos_label='positive', average='micro'))
print("F1 Score : ", f1_score(y_test, rf_model_pred,pos_label='positive', average='micro'))
print("Accuracy Score : ", accuracy_score(y_test, rf_model_pred))
```

```
Precision Score :  0.9224598930481284
Recall Score :  0.9224598930481284
F1 Score :  0.9224598930481284
Accuracy Score :  0.9224598930481284
```

Fig 18: Model Accuracy summary