-

|  |  |  |
| --- | --- | --- |
|  | |  |
| My NEAMusic Suggestion Tool | | |
| Abdulai Yusuf |  | |

|  |
| --- |
|  |
|  |
| Contents  [Investigation 4](#_Toc383253370)  [Introduction 4](#_Toc383253371)  [Questionnaires 8](#_Toc383253373)  [Interview 12](#_Toc383253374)  [Existing system documents 14](#_Toc383253376)  [Flowcharts 26](file:///C:/Users/Andrew/Dropbox/A2/Computing/Project/Project.docx#_Toc383253378)  [Analysis 42](#_Toc383253380)  [Scope and Objectives 42](#_Toc383253381)  [Tasks to be computerised 43](#_Toc383253382)  [Solutions 44](#_Toc383253383) |

Introduction

Music has many ways to affect people. In some ways, it is good for the body, both physically and mentally. People think it is an important thing to listen to in other ways. More detailed, music has personalities, which can express what people feel. There are many observations involving diverse ways to express human emotions. Emotions are exciting things, especially when they involve music. Music can have many personalities, affect people’s emotions, and be used as therapy.

If music reveals emotions, it is not a normal emotion. The expression in music can be considered a traditionalist phenomenon and can slow down when softer music plays, such as a lullaby. In addition, music rhythms can adjust brain waves and breathing patterns so that the vibrations from the music impact the body, which can change peoples’ moods and bodily functions.

Music has a good reputation, and it attracts people worldwide. However, people find it hard to find a song like a genre they listen to. The problem is worsened by people struggling to find their favourite song, their friend has a similar style, but they cannot always find their friends when they are busy.

Research in music recommender systems (MRS) has as of late met a generous increase in interest in the business. Because of Spotify, Pandora, or Apple Music, music specialists are given admittance to several millions of music pieces. By sifting this host of music things, so restricting decision over-burden, MRS are often extremely fruitful to propose melodies that fit the users’ intentions.

In any case, such frameworks are still a long way from being awesome and regularly produce unacceptable proposals. This is incomplete because users’ preferences are exceptionally subject to many elements, which are not considered in sufficient depth in current MRS draws near, which are customarily fixated on the central idea of client thing connections, or now and then substance-based thing descriptors.

Inversely, we contend that significant the users’ melodic amusement needs require considering the audience members’ fundamental, outward, and context-oriented parts, just as more respectable collaboration data. For example, the audience members’ energy, danceability, and happiness just as their actions affect musical preferences and requirements. So are users’ logical elements, including climate conditions, social encompassing, or places of interest.

Likewise, the piece and explanation of a music playlist or a listening meeting uncover data regarding which tunes go well together or are appropriate for a specific event. In this way, scientists and originators of MRS ought to rethink their clients in an all-encompassing way to construct frameworks custom fitted to the specificities of every client.

[Put more research here]

# Problem definition

#### Aim

This project aims to create a free music recommendation system using Python to estimate the user’s musical preferences and elaborate recommendations of several musical elements according to these preferences. For me to decide the age range of the group we will analyse the number of daily users on Spotify per age group:

The leading group to benefit from the new system will be people (aged 25–34). This will help them find the song quickly and feel more comfortable and relaxed in listening to more music of their type.

Interview

[Interviews go here – Questionnaires also go here. Interview some people and make results and percentages based on that]

# Interview Summary:

[bullet points go here]

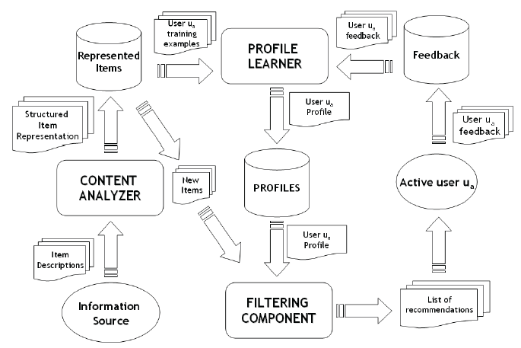
#### Summary of System

I will analyse the process used to automate results based on a users’ preference using a Spotify database.Scope and ObjectivesThe goals of this project are outlined below:

* Create a music recommendation system to infer the user’s musical preferences each time. The scope is to know some details about the user, really diving into the users’ preferences
* Explore the current music services, looking for a complete and freely accessible music catalogue and free streaming services.
* Develop a working system capable of making the most of free online services to provide the user with a completely free system that brings the opportunity of discovering recent music.

##### Recommendation Systems

# Content-Based Recommendation Systems

Content-Based Recommendation Systems or (CBRS) recommends items based on their features and the similarity between elements of other items. Assume a user has already seen a song from Pop Music; CBRS will recommend music that belongs to the Pop genre. Recommendation Systems have the effect of guiding users personally to intriguing objects in an ample space of feasible options. Content-based recommendation systems recommend items like those a user has liked in the past. Indeed, the basic process performed by a content-based recommender consists of matching up the attributes of a user profile in which preferences and interests are stored, with the attributes of a song, to recommend new interesting songs to the user.

[write a bit more about recommendation systems]

# Conclusion

This review shows the range of choices to create once designing a recommender system, giving an outline that eases the decision-making method upon analysis’ part. Some choices visualised once this analysis state that the user profile utilised during this recommender may be supported history-based generation, thanks to the watching capabilities of the IDE itself.

The user profile will be refreshed once new data is retrieved from interaction, so the user profile is endlessly evolving. The sole relevant feedback taken under consideration for this purpose is strictly implicit. It is retrieved optimally by the IDE.

These methods use an item profile within the system. To abstract the features of the items in the system. A widely used algorithm is the tf–idf representation.

Analysis

Solutions

[UML Diagrams go here]

[Other charts go here]

##### Requirements

1. Users must be able to identify a track in which will be analysed
   1. There must be a short selection of options in which the user will select
   2. The user must be able to type the song click ‘Search.’
      1. This operation can be run by clicking a button. Python will run the code, link Python with the million-song database and output the possible results
      2. There must be a sufficient amount of metadata within the track
         1. There must be an artist name for the track
         2. There must be a song title for the track
         3. There must be the year for the track in which a song is released
         4. There must be a genre for the track
         5. The track should have a sufficient length of music
         6. The track should have a tempo
2. The track must have a measure of how much energy, danceability and happiness a particular track has
3. There should be a measure from 0 to 1 on the probability that the other song is alike
   1. Probability must be visible and measurable using Pearson’s Correlation Coefficient
   2. Feedback – Detailed, immediate feedback given upon submission of code to the program administrators (myself).
4. Recommendation Systems should be used to find a top result for the desired search
5. It must be possible to sign in/sign out
   1. There will be a page for you to do so
6. It must then be possible to do another search
   1. The song that you recently searched for will be listed underneath as a reference

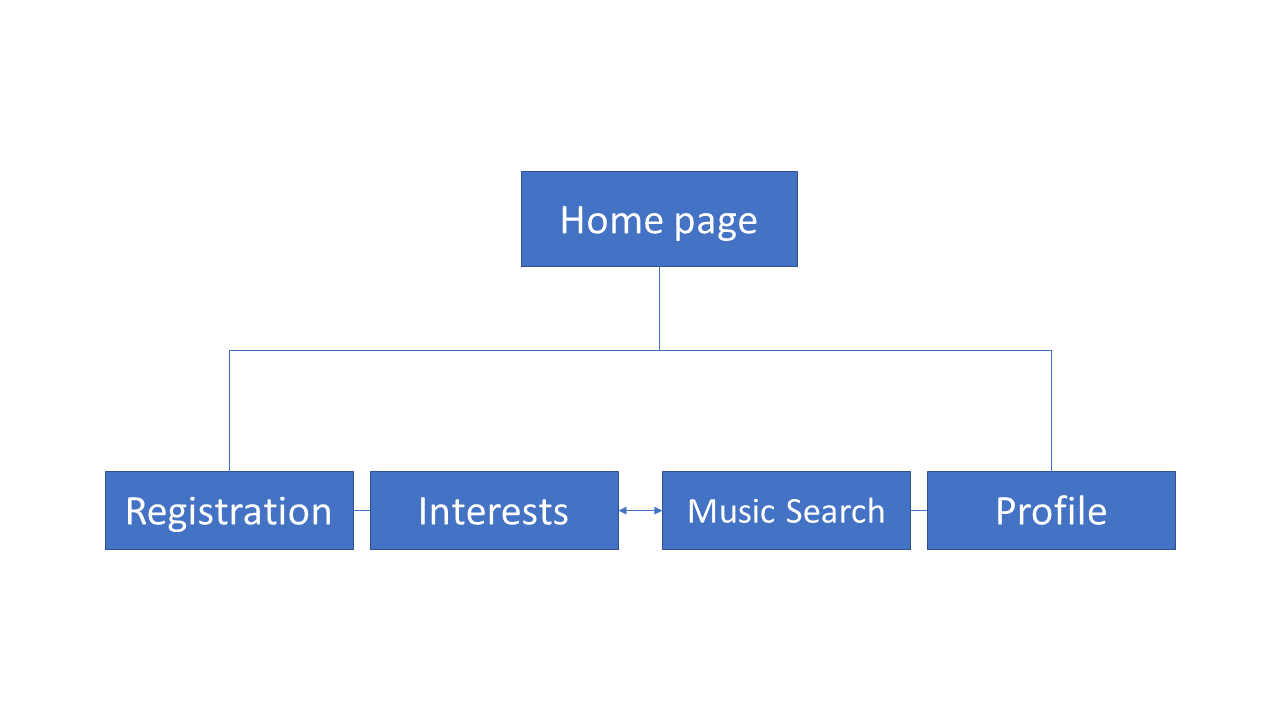
Design

Product overview

The objective of this project is to build an application using Python to deliver a music suggestion tool.

Tasks to be computerised

1. Unpractised users must register to use the system:
   * The new user must complete a registration tab in which they enter their name and email address
   * The new user should choose a user id and password
   * The user id can be unique
   * Passwords must be at least 8 characters long and hold a mixture of character types
2. To log in, the user must enter their user id and password.
   * After four unsuccessful login attempts, the user must be locked out of the system. They must then contact the administrator to get the account unlocked.



|  |  |  |
| --- | --- | --- |
| Data item | Data type | Validation/Reference |
| Name | String | Required |
| Email | String |  |
| User ID | String |  |
| Password | String |  |
| Song ID | String |  |
| Listen Count | Real |  |
| Artist name | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Process | Storage | Output |
| Name  Email  User ID  Password | Validate input  Secure password  Generate email | User table (database) | A message stating a secure password  Secured  password |
| Song ID  Listen Count  Artist Name  Title  Year Release | Validate input  Generate song data  Generate similar music | Million Song Dataset (database) | A table showing the list of similar songs alike |
| Percentage | Assign percentage as a probability from 0 to 1  Use Pearson’s Correlation Coefficient using the song data as a reference to measure the correlation of songs and their style. | Million Song Dataset (database) | A table showing the list of similar songs alike |

[Proper navigation diagram]

#### Navigation at a glance:

[Screenshot of your project step by step with notes on simplifying the readers understanding of the pages]

# Database design

Entity-Relationship diagram

SQL Statements

[Put SQL Statements of your interface]

##### Proof of Concept

Registration Page:

from tkinter import \*

from tkinter import messagebox

import sqlite3

f = ("Calibri", 14)

con = sqlite3.connect('userdata.db')

cur = con.cursor()

cur.execute('''CREATE TABLE IF NOT EXISTS record(

name text,

email text,

contact number,

gender text,

country text,

password text

)

''')

con.commit()

ws = Tk()

ws.title('Registration')

ws.geometry('940x500')

ws.config(bg='#0bd67b')

def insert\_record():

check\_counter=0

warn = ""

if register\_name.get() == "":

warn = "Name can't be empty"

else:

check\_counter += 1

if register\_email.get() == "":

warn = "Email can't be empty"

else:

check\_counter += 1

if register\_mobile.get() == "":

warn = "Contact can't be empty"

else:

check\_counter += 1

if var.get() == "":

warn = "Gender"

else:

check\_counter += 1

if variable.get() == "":

warn = "Select Country"

else:

check\_counter += 1

if register\_pwd.get() == "":

warn = "Password can't be empty"

else:

check\_counter += 1

if pwd\_again.get() == "":

warn = "Re-enter password can't be empty"

else:

check\_counter += 1

if register\_pwd.get() != pwd\_again.get():

warn = "Passwords didn't match!"

else:

check\_counter += 1

if check\_counter == 8:

try:

con = sqlite3.connect('userdata.db')

cur = con.cursor()

cur.execute("INSERT INTO record VALUES (:name, :email, :contact, :gender, :country, :password)", {

'name': register\_name.get(),

'email': register\_email.get(),

'contact': register\_mobile.get(),

'gender': var.get(),

'country': variable.get(),

'password': register\_pwd.get()

})

con.commit()

messagebox.showinfo('confirmation', 'Record Saved')

except Exception as ep:

messagebox.showerror('', ep)

else:

messagebox.showerror('Error', warn)

def login\_response():

try:

con = sqlite3.connect('userdata.db')

c = con.cursor()

for row in c.execute("Select \* from record"):

username = row[1]

pwd = row[5]

except Exception as ep:

messagebox.showerror('', ep)

uname = email\_tf.get()

upwd = pwd\_tf.get()

check\_counter=0

if uname == "":

warn = "Username can't be empty"

else:

check\_counter += 1

if upwd == "":

warn = "Password can't be empty"

else:

check\_counter += 1

if check\_counter == 2:

if (uname == username and upwd == pwd):

messagebox.showinfo('Login Status', 'Logged in Successfully!')

else:

messagebox.showerror('Login Status', 'invalid username or password')

else:

messagebox.showerror('', warn)

var = StringVar()

var.set('male')

countries = []

variable = StringVar()

world = "United Kingdom"

for country in world:

country = country.rstrip('\n')

countries.append(country)

variable.set(countries[1])

# widgets

left\_frame = Frame(

ws,

bd=2,

bg='#CCCCCC',

relief=SOLID,

padx=10,

pady=10

)

Label(

left\_frame,

text="Enter Email",

bg='#CCCCCC',

font=f).grid(row=0, column=0, sticky=W, pady=10)

Label(

left\_frame,

text="Enter Password",

bg='#CCCCCC',

font=f

).grid(row=1, column=0, pady=10)

email\_tf = Entry(

left\_frame,

font=f

)

pwd\_tf = Entry(

left\_frame,

font=f,

show='\*'

)

login\_btn = Button(

left\_frame,

width=15,

text='Login',

font=f,

relief=SOLID,

cursor='hand2',

command=login\_response

)

right\_frame = Frame(

ws,

bd=2,

bg='#a1b5a3',

relief=SOLID,

padx=10,

pady=10

)

Label(

right\_frame,

text="Enter Name",

bg='#a1b5a3',

font=f

).grid(row=0, column=0, sticky=W, pady=10)

Label(

right\_frame,

text="Enter Email",

bg='#a1b5a3',

font=f

).grid(row=1, column=0, sticky=W, pady=10)

Label(

right\_frame,

text="Contact Number",

bg='#a1b5a3',

font=f

).grid(row=2, column=0, sticky=W, pady=10)

Label(

right\_frame,

text="Select Gender",

bg='#a1b5a3',

font=f

).grid(row=3, column=0, sticky=W, pady=10)

Label(

right\_frame,

text="Select Country",

bg='#a1b5a3',

font=f

).grid(row=4, column=0, sticky=W, pady=10)

Label(

right\_frame,

text="Enter Password",

bg='#a1b5a3',

font=f

).grid(row=5, column=0, sticky=W, pady=10)

Label(

right\_frame,

text="Re-Enter Password",

bg='#a1b5a3',

font=f

).grid(row=6, column=0, sticky=W, pady=10)

gender\_frame = LabelFrame(

right\_frame,

bg='#a1b5a3',

padx=10,

pady=10,

)

register\_name = Entry(

right\_frame,

font=f

)

register\_email = Entry(

right\_frame,

font=f

)

register\_mobile = Entry(

right\_frame,

font=f

)

male\_rb = Radiobutton(

gender\_frame,

text='Male',

bg='#a1b5a3',

variable=var,

value='male',

font=('Times', 10),

)

female\_rb = Radiobutton(

gender\_frame,

text='Female',

bg='#a1b5a3',

variable=var,

value='female',

font=('Times', 10),

)

others\_rb = Radiobutton(

gender\_frame,

text='Others',

bg='#a1b5a3',

variable=var,

value='others',

font=('Times', 10)

)

register\_country = OptionMenu(

right\_frame,

variable,

\*countries)

register\_country.config(

width=15,

font=('Times', 12)

)

register\_pwd = Entry(

right\_frame,

font=f,

show='\*'

)

pwd\_again = Entry(

right\_frame,

font=f,

show='\*'

)

register\_btn = Button(

right\_frame,

width=15,

text='Register',

font=f,

relief=SOLID,

cursor='hand2',

command=insert\_record

)

# widgets placement

email\_tf.grid(row=0, column=1, pady=10, padx=20)

pwd\_tf.grid(row=1, column=1, pady=10, padx=20)

login\_btn.grid(row=2, column=1, pady=10, padx=20)

left\_frame.place(x=50, y=50)

register\_name.grid(row=0, column=1, pady=10, padx=20)

register\_email.grid(row=1, column=1, pady=10, padx=20)

register\_mobile.grid(row=2, column=1, pady=10, padx=20)

register\_country.grid(row=4, column=1, pady=10, padx=20)

register\_pwd.grid(row=5, column=1, pady=10, padx=20)

pwd\_again.grid(row=6, column=1, pady=10, padx=20)

register\_btn.grid(row=7, column=1, pady=10, padx=20)

right\_frame.place(x=500, y=50)

gender\_frame.grid(row=3, column=1, pady=10, padx=20)

male\_rb.pack(expand=True, side=LEFT)

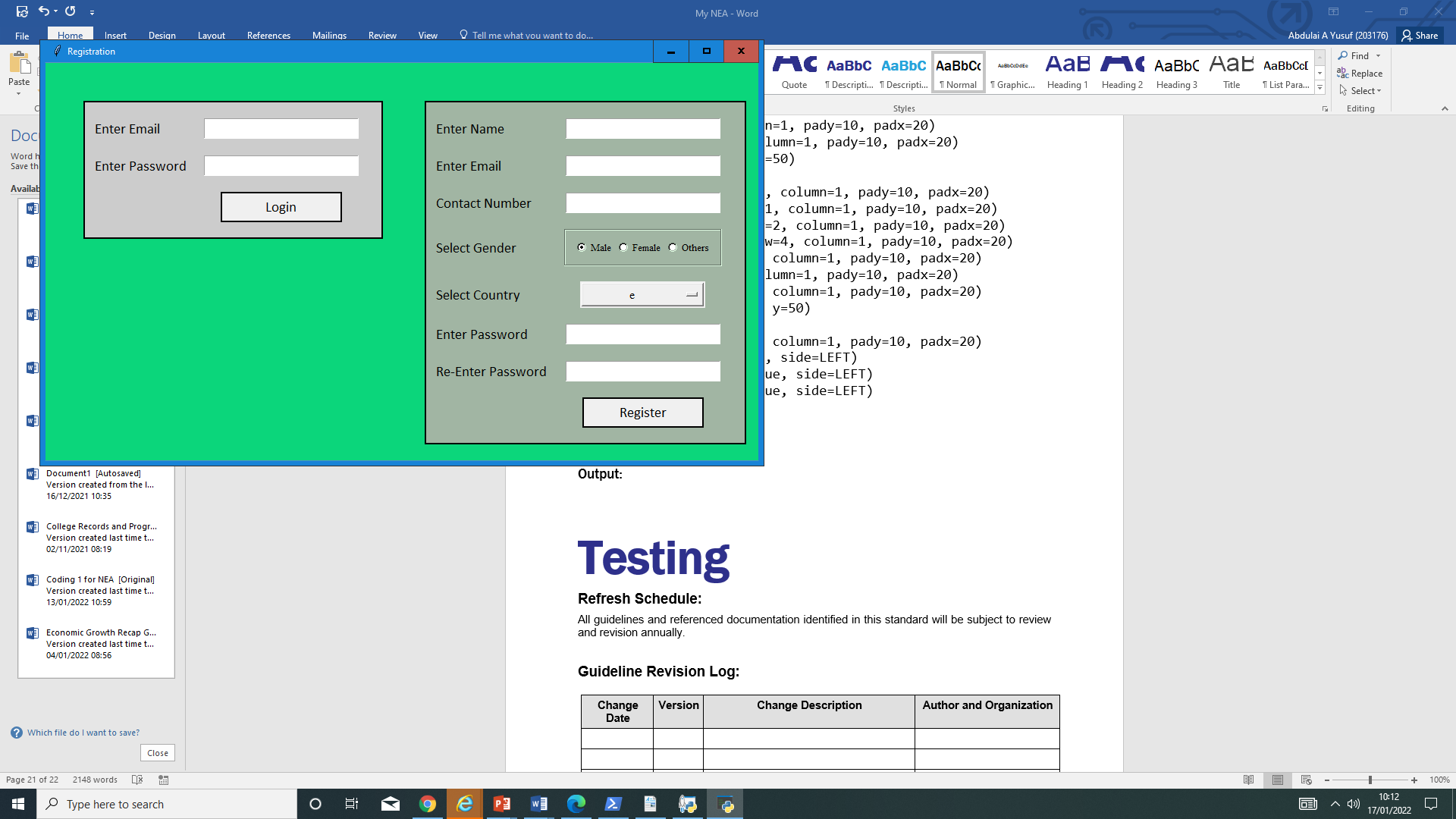
female\_rb.pack(expand=True, side=LEFT)

others\_rb.pack(expand=True, side=LEFT)

# infinite loop

ws.mainloop()

Output:



Testing

**Refresh Schedule:**

All guidelines and referenced documentation identified in this standard will be subject to review and revision annually.

**Guideline Revision Log:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Change Date** | **Version** | **Change Description** | **Author and Organization** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

PYQT