Import re

Import motor.motor\_asyncio # pylint: disable=import-error

From bot import DB\_URI # pylint: disable=import-error

Class Singleton(type):

\_\_instances\_\_ = {}

Def \_\_call\_\_(cls, \*args, \*\*kwargs):

If cls not in cls.\_\_instances\_\_:

Cls.\_\_instances\_\_[cls] = super(Singleton, cls).\_\_call\_\_(\*args, \*\*kwargs)

Return cls.\_\_instances\_\_[cls]

Class Database(metaclass=Singleton):

Def \_\_init\_\_(self):

Self.\_client = motor.motor\_asyncio.AsyncIOMotorClient(DB\_URI)

Self.db = self.\_client[“Adv\_Auto\_Filter”]

Self.col = self.db[“Main”]

Self.acol = self.db[“Active\_Chats”]

Self.fcol = self.db[“Filter\_Collection”]

Self.cache = {}

Self.acache = {}

Async def create\_index(self):

“””

Create text index if not in db

“””

Await self.fcol.create\_index([(“file\_name”, “text”)])

Def new\_chat(self, group\_id, channel\_id, channel\_name):

“””

Create a document in db if the chat is new

“””

Try:

Group\_id, channel\_id = int(group\_id), int(channel\_id)

Except:

Pass

Return dict(

\_id = group\_id,

Chat\_ids = [{

“chat\_id”: channel\_id,

“chat\_name”: channel\_name

}],

Types = dict(

Audio=False,

Document=True,

Video=True

),

Configs = dict(

Accuracy=0.80,

Max\_pages=5,

Max\_results=50,

Max\_per\_page=10,

Pm\_fchat=True,

Show\_invite\_link=True

)

)

Async def status(self, group\_id: int):

“””

Get the total filters, total connected

Chats and total active chats of a chat

“””

Group\_id = int(group\_id)

Total\_filter = await self.tf\_count(group\_id)

Chats = await self.find\_chat(group\_id)

Chats = chats.get(“chat\_ids”)

Total\_chats = len(chats) if chats is not None else 0

Achats = await self.find\_active(group\_id)

If achats not in (None, False):

Achats = achats.get(“chats”)

If achats == None:

Achats = []

Else:

Achats = []

Total\_achats = len(achats)

Return total\_filter, total\_chats, total\_achats

Async def find\_group\_id(self, channel\_id: int):

“””

Find all group id which is connected to a channel

For add a new files to db

“””

Data = self.col.find({})

Group\_list = []

For group\_id in await data.to\_list(length=50): # No Need Of Even 50

For y in group\_id[“chat\_ids”]:

If int(y[“chat\_id”]) == int(channel\_id):

Group\_list.append(group\_id[“\_id”])

Else:

Continue

Return group\_list

# Related TO Finding Channel(s)

Async def find\_chat(self, group\_id: int):

“””

A funtion to fetch a group’s settings

“””

Connections = self.cache.get(str(group\_id))

If connections is not None:

Return connections

Connections = await self.col.find\_one({‘\_id’: group\_id})

If connections:

Self.cache[str(group\_id)] = connections

Return connections

Else:

Return self.new\_chat(None, None, None)

Async def add\_chat(self, group\_id: int, channel\_id: int, channel\_name):

“””

A funtion to add/update a chat document when a new chat is connected

“””

New = self.new\_chat(group\_id, channel\_id, channel\_name)

Update\_d = {“$push” : {“chat\_ids” : {“chat\_id”: channel\_id, “chat\_name” : channel\_name}}}

Prev = await self.col.find\_one({‘\_id’:group\_id})

If prev:

Await self.col.update\_one({‘\_id’:group\_id}, update\_d)

Await self.update\_active(group\_id, channel\_id, channel\_name)

Await self.refresh\_cache(group\_id)

Return True

Self.cache[str(group\_id)] = new

Await self.col.insert\_one(new)

Await self.add\_active(group\_id, channel\_id, channel\_name)

Await self.refresh\_cache(group\_id)

Return True

Async def del\_chat(self, group\_id: int, channel\_id: int):

“””

A Funtion to delete a channel and its files from db of a chat connection

“””

Group\_id, channel\_id = int(group\_id), int(channel\_id) # group\_id and channel\_id Didn’t type casted to int for some reason

Prev = self.col.find\_one({“\_id”: group\_id})

If prev:

Await self.col.update\_one(

{“\_id”: group\_id},

{“$pull” :

{“chat\_ids” :

{“chat\_id”:

Channel\_id

}

}

},

False,

True

)

Await self.del\_active(group\_id, channel\_id)

Await self.refresh\_cache(group\_id)

Return True

Return False

Async def in\_db(self, group\_id: int, channel\_id: int):

“””

Check whether if the given channel id is in db or not…

“””

Connections = self.cache.get(group\_id)

If connections is None:

Connections = await self.col.find\_one({‘\_id’: group\_id})

Check\_list = []

If connections:

For x in connections[“chat\_ids”]:

Check\_list.append(int(x.get(“chat\_id”)))

If int(channel\_id) in check\_list:

Return True

Return False

Async def update\_settings(self, group\_id: int, settings):

“””

A Funtion to update a chat’s filter types in db

“””

Group\_id = int(group\_id)

Prev = await self.col.find\_one({“\_id”: group\_id})

If prev:

Try:

Await self.col.update\_one({“\_id”: group\_id}, {“$set”: {“types”: settings}})

Await self.refresh\_cache(group\_id)

Return True

Except Exception as e:

Print €

Return False

Print(“You Should First Connect To A Chat To Use This Funtion….. ‘databse.py/#201’ “)

Return False

Async def update\_configs(self, group\_id: int, configs):

“””

A Funtion to update a chat’s configs in db

“””

Prev = await self.col.find\_one({“\_id”: group\_id})

If prev:

Try:

Await self.col.update\_one(prev, {“$set”:{“configs”: configs}})

Await self.refresh\_cache(group\_id)

Return True

Except Exception as e:

Print €

Return False

Print(“You Should First Connect To A Chat To Use This”)

Return False

Async def delete\_all(self, group\_id: int):

“””

A Funtion to delete all documents related to a

Chat from db

“””

Prev = await self.col.find\_one({“\_id”: group\_id})

If prev:

Await self.delall\_active(group\_id)

Await self.delall\_filters(group\_id)

Await self.del\_main(group\_id)

Await self.refresh\_cache(group\_id)

Return

Async def del\_main(self, group\_id: int):

“””

A Funtion To Delete the chat’s main db document

“””

Await self.col.delete\_one({“\_id”: group\_id})

Await self.refresh\_cache(group\_id)

Return True

Async def refresh\_cache(self, group\_id: int):

“””

A Funtion to refresh a chat’s chase data

In case of update in db

“””

If self.cache.get(str(group\_id)):

Self.cache.pop(str(group\_id))

Prev = await self.col.find\_one({“\_id”: group\_id})

If prev:

Self.cache[str(group\_id)] = prev

Return True

# Related To Finding Active Channel(s)

Async def add\_active(self, group\_id: int, channel\_id: int, channel\_name):

“””

A Funtion to add a channel as an active chat the a connected group

(This Funtion will be used only if it’s the first time)

“””

Templ = {“\_id”: group\_id, “chats”:[{“chat\_id”: channel\_id, “chat\_name”: channel\_name}]}

Try:

Await self.acol.insert\_one(templ)

Await self.refresh\_acache(group\_id)

Except Exception as e:

Print€

Return False

Return True

Async def del\_active(self, group\_id: int, channel\_id: int):

“””

A funtion to delete a channel from active chat colletion in db

“””

Templ = {“$pull”: {“chats”: dict(chat\_id = channel\_id)}}

Try:

Await self.acol.update\_one({“\_id”: group\_id}, templ, False, True)

Except Exception as e:

Print€

Pass

Await self.refresh\_acache(group\_id)

Return True

Async def update\_active(self, group\_id: int, channel\_id: int, channel\_name):

“””

A Funtion to add a new active chat to the connected group

“””

Group\_id, channel\_id = int(group\_id), int(channel\_id)

Prev = await self.acol.find\_one({“\_id”: group\_id})

Templ = {“$push” : {“chats” : dict(chat\_id = channel\_id, chat\_name = channel\_name)}}

In\_c = await self.in\_active(group\_id, channel\_id)

If prev:

If not in\_c:

Await self.acol.update\_one({“\_id”: group\_id}, templ)

Else:

Return False

Else:

Await self.add\_active(group\_id, channel\_id, channel\_name)

Return True

Async def find\_active(self, group\_id: int):

“””

A Funtion to find all active chats of

A group from db

“””

If self.acache.get(str(group\_id)):

Self.acache.get(str(group\_id))

Connection = await self.acol.find\_one({“\_id”: group\_id})

If connection:

Self.acache[str(group\_id)] = connection

Return connection

Return False

Async def in\_active(self, group\_id: int, channel\_id: int):

“””

A Funtion to check if a chat id is in the active

Chat id list in db

“””

Prev = await self.acol.find\_one({“\_id”: group\_id})

If prev:

For x in prev[“chats”]:

If x[“chat\_id”] == channel\_id:

Return True

Return False

Return False

Async def delall\_active(self, group\_id: int):

“””

A Funtion to Delete all active chats of

A group from db

“””

Await self.acol.delete\_one({“\_id”:int(group\_id)})

Await self.refresh\_acache(group\_id)

Return

Async def refresh\_acache(self, group\_id: int):

“””

A Funtion to refresh a active chat’s chase data

In case of update in db

“””

If self.acache.get(str(group\_id)):

Self.acache.pop(str(group\_id))

Prev = await self.acol.find\_one({“\_id”: group\_id})

If prev:

Self.acache[str(group\_id)] = prev

Return True

# Related To Finding Filter(s)

Async def add\_filters(self, data):

“””

A Funtion to add document as

A bulk to db

“””

Try:

Await self.fcol.insert\_many(data)

Except Exception as e:

Print€

Return True

Async def del\_filters(self, group\_id: int, channel\_id: int):

“””

A Funtion to delete all filters of a specific

Chat and group from db

“””

Group\_id, channel\_id = int(group\_id), int(channel\_id)

Try:

Await self.fcol.delete\_many({“chat\_id”: channel\_id, “group\_id”: group\_id})

Print(await self.cf\_count(group\_id, channel\_id))

Return True

Except Exception as e:

Print€

Return False

Async def delall\_filters(self, group\_id: int):

“””

A Funtion To delete all filters of a group

“””

Await self.fcol.delete\_many({“group\_id”: int(group\_id)})

Return True

Async def get\_filters(self, group\_id: int, keyword: str):

“””

A Funtion to fetch all similar results for a keyowrd

From using text index

“””

Achats = await self.find\_active(group\_id)

Achat\_ids=[]

If not achats:

Return False

For chats in achats[“chats”]:

Achat\_ids.append(chats.get(“chat\_id”))

Filters = []

Pattern = keyword.lower().strip().replace(‘ ‘,’.\*’)

Raw\_pattern = r”\b{}\b”.format(pattern)

Regex = re.compile(raw\_pattern, flags=re.IGNORECASE)

Db\_list = self.fcol.find({“group\_id”: group\_id,”file\_name”: regex})

For document in await db\_list.to\_list(length=600):

If document[“chat\_id”] in achat\_ids:

Filters.append(document)

Else:

Continue

Return filters

Async def get\_file(self, unique\_id: str):

“””

A Funtion to get a specific files using its

Unique id

“””

File = await self.fcol.find\_one({“unique\_id”: unique\_id})

File\_id = None

File\_type = None

File\_name = None

File\_caption = None

If file:

File\_id = file.get(“file\_id”)

File\_name = file.get(“file\_name”)

File\_type = file.get(“file\_type”)

File\_caption = file.get(“caption”)

Return file\_id, file\_name, file\_caption, file\_type

Async def cf\_count(self, group\_id: int, channel\_id: int):

“””

A Funtion To count number of filter in channel

w.r.t the connect group

“””

Return await self.fcol.count\_documents({“chat\_id”: channel\_id, “group\_id”: group\_id})

Async def tf\_count(self, group\_id: int):

“””

A Funtion to count total filters of a group

“””

Return await self.fcol.count\_documents({“group\_id”: group\_id})