Gardner API Utility Documentation

1.2.0

Willem van der Schans, DoxyGen

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 AuthUtil.AuthUtil Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- StandardStatus
- ListedOrModified
- file_name
- append_file
- keyPath
- filePath
- k
- keyFlag
- jsonDict
- passFlagUre
- passFlagCm
- outcomeText
- popupFlag

Private Member Functions

- def __SetValues (self, values)
- def __ShowGui (self, layout, text)
- def __CreateFrame (self)

3.1.1 Detailed Description

Definition at line 18 of file AuthUtil.py.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 __init__()

```
def AuthUtil.AuthUtil.__init__ (
              self )
```

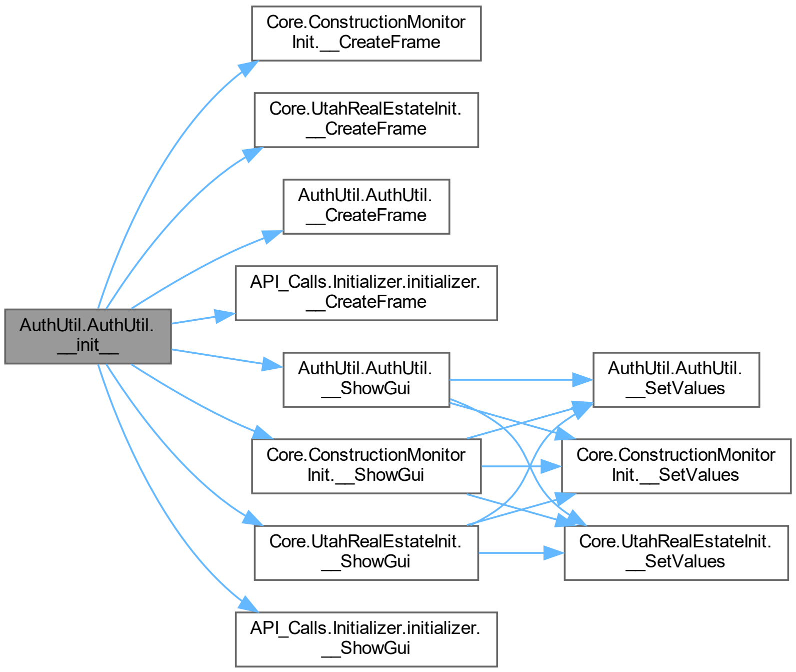
```
The __init__ function is called when the class is instantiated.
It sets up the initial state of the object, which in this case means that it creates a new window and displays it

Args:
self: Represent the instance of the class

Returns:
None

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 20 of file AuthUtil.py.

```
00020     def __init__(self):
00021 
00022         """
00023     The __init__ function is called when the class is instantiated.
00024     It sets up the initial state of the object, which in this case means that it creates a new window and
      displays it on screen.
00025 
00026     Args:
00027         self: Represent the instance of the class
00028 
00029     Returns:
00030         None
00031 
00032     Doc Author:
00033         Willem van der Schans, Trelent AI
00034     """
00035         self.StandardStatus = None
00036         self.ListedOrModified = None
00037         self.file_name = None
00038         self.append_file = None
00039         self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
00040         self.filePath =
      Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath("Security")
00041         self.k = None
00042         self.keyFlag = True
00043         self.jsonDict = {}
00044         self.passFlagUre = False
00045         self.passFlagCm = False
00046         self.outcomeText = "Please input the plain text keys in the input boxes above \n " \
00047                            "Submitting will overwrite any old values in an unrecoverable manner."
00048 
00049         if os.path.exists(self.filePath):
00050             pass
00051         else:
00052             if os.path.exists(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData")):
00053                 os.mkdir(self.filePath)
00054             else:
00055                 os.mkdir(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData"))
00056                 os.mkdir(self.filePath)
00057 
00058         if os.path.exists(self.keyPath):
00059             pass
00060         else:
00061             if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00062                 os.mkdir(self.keyPath)
00063             else:
```

```
                os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
                os.mkdir(self.keyPath)

        if os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
            try:
                f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                self.k = f.readline()
                f.close()
            except Exception as e:
                print(e)
                RESTError(402)
                raise SystemExit(402)
        else:
            self.k = Fernet.generate_key()
            f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
            f.write(self.k)
            f.close()

            try:
                os.remove(self.filePath.joinpath("auth.json"))
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |
Error = {e} | Error in removing auth.json file - This can be due to the file not existing. Continuing...")
                pass

            f = open(self.filePath.joinpath("auth.json"), "wb")
            f.close()
            self.keyFlag = False

        self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")

        try:

ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), 2)
        except Exception as e:
            # Logging
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error =
{e} | Error when setting the key file as hidden. This is either a Permission error or Input Error.
Continuing...")
            pass

```

Here is the call graph for this function:


Core.ConstructionMonitor Init.__CreateFrame
Core.UtahRealEstateInit. __CreateFrame
AuthUtil.AuthUtil. __CreateFrame
API_Calls.Initializer.initializer. __CreateFrame
AuthUtil.AuthUtil. __init__
AuthUtil.AuthUtil. __ShowGui
AuthUtil.AuthUtil. __SetValues
Core.ConstructionMonitor Init.__ShowGui
Core.ConstructionMonitor Init.__SetValues
Core.UtahRealEstateInit. __ShowGui
Core.UtahRealEstateInit. __SetValues
API_Calls.Initializer.initializer. __ShowGui

3.1.3 Member Function Documentation

3.1.3.1 __CreateFrame()

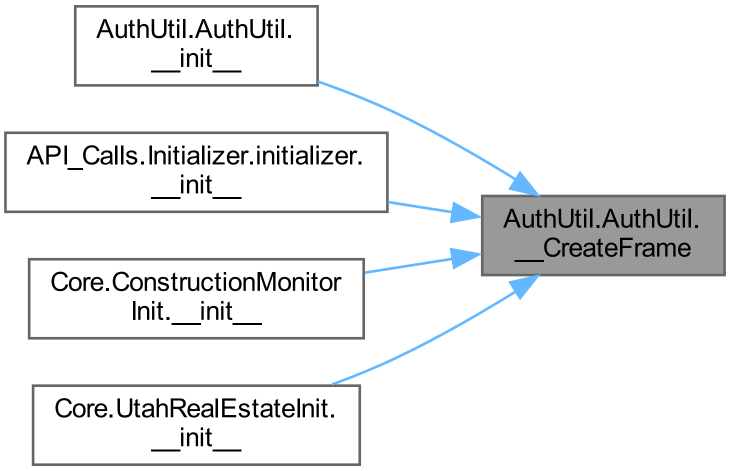
```
def AuthUtil.AuthUtil.__CreateFrame (
              self )  [private]
```

```
The __CreateFrame function creates the GUI layout for the Authentication Utility.
It is called by __init__ and returns a list of lists that contains all the elements
that will be displayed in the window.

Args:
self: Access the class attributes and methods

Returns:
A list of lists

Doc Author:
Trelent
```

Definition at line 235 of file AuthUtil.py.

```
    def __CreateFrame(self):
        """
    The __CreateFrame function creates the GUI layout for the Authentication Utility.
    It is called by __init__ and returns a list of lists that contains all the elements
    that will be displayed in the window.

    Args:
        self: Access the class attributes and methods

    Returns:
        A list of lists

    Doc Author:
        Trelent
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"),
justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line2 = [sg.Push(),
                 sg.Text("Utah Real Estate API Key: ", justification="center"),
                 sg.Push()]

        line3 = [sg.Push(),
                 sg.Input(default_text="123", key="-ureAuth-", disabled=False,
                          size=(40, 1)),
                 sg.Push()]

        line4 = [sg.HSeparator()]

        line5 = [sg.Push(),
                 sg.Text("Construction Monitor HTTP BASIC Key: ", justification="center"),
                 sg.Push()]

        line6 = [sg.Push(),
                 sg.Input(default_text="Basic 123", key="-cmAuth-", disabled=False,
                          size=(40, 1)),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Push(),
                 sg.Text(self.outcomeText, justification="center"),
                 sg.Push()]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout
```

Here is the caller graph for this function:



3.1.3.2 __SetValues()

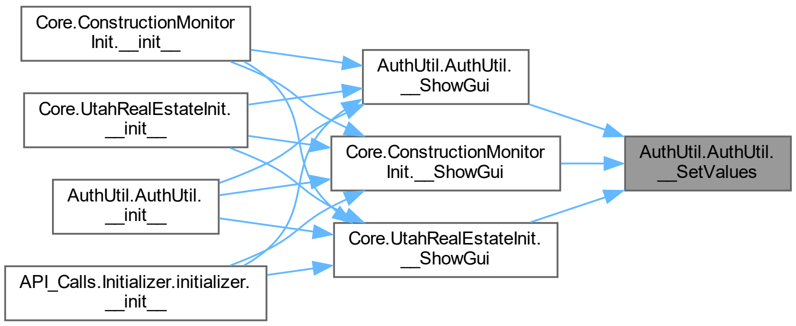
```
def AuthUtil.AuthUtil.__SetValues (
              self,
              values )  [private]
```

```
The __SetValues function is called when the user clicks on the &quot;OK&quot; button in the window.
It takes a dictionary of values as an argument, and then uses those values to update
the auth.json file with new keys for both Utah Real Estate and Construction Monitor.

Args:
self: Make the function a method of the class
values: Store the values that are entered into the form

Returns:
A dictionary of the values entered by the user

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 104 of file AuthUtil.py.

```
    def __SetValues(self, values):

        """
    The __SetValues function is called when the user clicks on the &quot;OK&quot; button in the window.
    It takes a dictionary of values as an argument, and then uses those values to update
    the auth.json file with new keys for both Utah Real Estate and Construction Monitor.

    Args:
        self: Make the function a method of the class
        values: Store the values that are entered into the form

    Returns:
        A dictionary of the values entered by the user

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        ureCurrent = None
        cmCurrent = None
        keyFile = None
        self.popupFlag = False

        fernet = Fernet(self.k)

        try:
            f = open(self.filePath.joinpath("auth.json"), "r")
            keyFile = json.load(f)
            fileFlag = True
        except:
            fileFlag = False

        # Try initial decoding, if fails pass and write new keys and files
        if fileFlag:
            try:
                ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Utah Real Estate Key. Continuing but this should be resolved if URE functionality will be accessed")
                ureCurrent = None

            try:
                cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Construction Monitor Key. Continuing but this should be resolved if CM functionality will be accessed")
                cmCurrent = None

        if values["-ureAuth-"] != "":
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
            self.passFlagUre = True
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(ureCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if values["-cmAuth-"] != "":
            if values["-cmAuth-"].startswith("Basic"):
                self.jsonDict.update(
                    {"cm": {"parameter": "Authorization",
                            "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
                self.passFlagCm = True
            else:
                PopupWrapped("Please make sure you provide a HTTP Basic Auth key for construction Monitor",
                             windowType="AuthError")
                self.popupFlag = True
                pass
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"cm": {"parameter": "Authorization", "auth": fernet.encrypt(cmCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if not self.passFlagUre and not self.passFlagCm:
            PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction
Monitor",
                        windowType="errorLarge")
        if self.passFlagCm and not self.passFlagUre:
            PopupWrapped("Please make sure you provide a key for Utah Real estate",
windowType="errorLarge")
        if not self.passFlagCm and self.passFlagUre and not self.popupFlag:
            PopupWrapped("Please make sure you provide a key for Construction Monitor",
windowType="errorLarge")
        if self.popupFlag:
            pass
        else:
            jsonOut = json.dumps(self.jsonDict, indent=4)
            f = open(self.filePath.joinpath("auth.json"), "w")
            f.write(jsonOut)

```

Here is the caller graph for this function:


Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
AuthUtil.AuthUtil.__ShowGui
Core.ConstructionMonitorInit.__ShowGui
Core.UtahRealEstateInit.__ShowGui
AuthUtil.AuthUtil.__SetValues

3.1.3.3 __ShowGui()

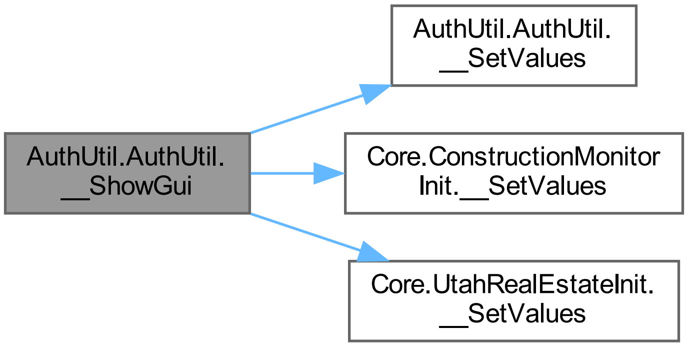
```
def AuthUtil.AuthUtil.__ShowGui (
              self,
              layout,
              text ) [private]
```

```
The __ShowGui function is a helper function that displays the GUI to the user.
It takes in two arguments: layout and text. The layout argument is a list of lists,
which contains all the elements that will be displayed on screen. The text argument
is simply what will be displayed at the top of the window.

Args:
self: Represent the instance of the class
layout: Pass the layout of the gui to be displayed
text: Set the title of the window

Returns:
A window object
```

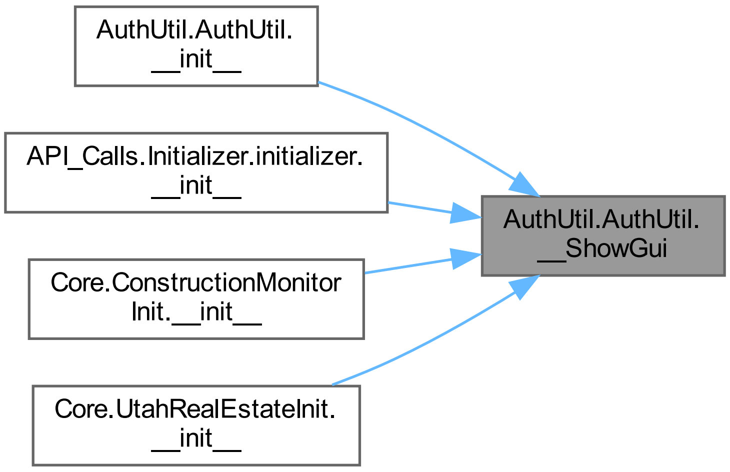
Definition at line 196 of file AuthUtil.py.

```
    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is a helper function that displays the GUI to the user.
    It takes in two arguments: layout and text. The layout argument is a list of lists,
    which contains all the elements that will be displayed on screen. The text argument
    is simply what will be displayed at the top of the window.

    Args:
        self: Represent the instance of the class
        layout: Pass the layout of the gui to be displayed
        text: Set the title of the window

    Returns:
        A window object
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while not self.passFlagUre or not self.passFlagCm:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                except Exception as e:
                    print(e)
                    RESTError(993)
                finally:
                    pass
            elif event == sg.WIN_CLOSED or event == "Quit":

                break
            else:
                pass

        window.close()

```

Here is the call graph for this function:


AuthUtil.AuthUtil.__ShowGui
AuthUtil.AuthUtil.__SetValues
Core.ConstructionMonitorInit.__SetValues
Core.UtahRealEstateInit.__SetValues

Here is the caller graph for this function:



3.1.4 Member Data Documentation

3.1.4.1 append_file

```
AuthUtil.AuthUtil.append_file
```

Definition at line 38 of file AuthUtil.py.

3.1.4.2 file_name

```
AuthUtil.AuthUtil.file_name
```

Definition at line 37 of file AuthUtil.py.

3.1.4.3 filePath

```
AuthUtil.AuthUtil.filePath
```

Definition at line 40 of file AuthUtil.py.

3.1.4.4 jsonDict

```
AuthUtil.AuthUtil.jsonDict
```

Definition at line 43 of file AuthUtil.py.

3.1.4.5 k

```
AuthUtil.AuthUtil.k
```

Definition at line 41 of file AuthUtil.py.

3.1.4.6 keyFlag

```
AuthUtil.AuthUtil.keyFlag
```

Definition at line 42 of file AuthUtil.py.

3.1.4.7 keyPath

```
AuthUtil.AuthUtil.keyPath
```

Definition at line 39 of file AuthUtil.py.

3.1.4.8 ListedOrModified

```
AuthUtil.AuthUtil.ListedOrModified
```

Definition at line 36 of file AuthUtil.py.

3.1.4.9 outcomeText

`AuthUtil.AuthUtil.outcomeText`

Definition at line 46 of file AuthUtil.py.

3.1.4.10 passFlagCm

`AuthUtil.AuthUtil.passFlagCm`

Definition at line 45 of file AuthUtil.py.

3.1.4.11 passFlagUre

`AuthUtil.AuthUtil.passFlagUre`

Definition at line 44 of file AuthUtil.py.

3.1.4.12 popupFlag

`AuthUtil.AuthUtil.popupFlag`

Definition at line 124 of file AuthUtil.py.

3.1.4.13 StandardStatus

`AuthUtil.AuthUtil.StandardStatus`

Definition at line 35 of file AuthUtil.py.

The documentation for this class was generated from the following file:

- AuthUtil.py

3.2 BatchProcessing.BatchProcessorConstructionMonitor Class Reference

Public Member Functions

- def __init__ (self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject)
- def FuncSelector (self)
- def ConstructionMonitorProcessor (self, valueObject)

Public Attributes

- dataframe
- valueObject

Private Attributes

- __numBatches
- __parameterDict
- __restDomain
- __headerDict
- __columnSelection
- __maxRequests
- __requestCount
- __requestCalls
- __dateTracker

3.2.1 Detailed Description

Definition at line 41 of file BatchProcessing.py.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 __init__()

```
def BatchProcessing.BatchProcessorConstructionMonitor.__init__ (
              self,
              RestDomain,
              NumBatches,
              ParameterDict,
              HeaderDict,
              ColumnSelection,
              valueObject )
```

```
The __init__ function is the constructor for a class. It is called when an object of that class
is created, and it sets up the attributes of that object. In this case, we are setting up our
object to have a dataframe attribute (which will be used to store all of our data), as well as
attributes for each parameter in our ReST call.

Args:
self: Represent the instance of the class
RestDomain: Specify the domain of the rest api
NumBatches: Determine how many batches of data to retrieve
ParameterDict: Pass in the parameters that will be used to make the api call
HeaderDict: Pass the header dictionary from the main function to this class
ColumnSelection: Determine which columns to pull from the api
valueObject: Pass in the value object that is used to determine what values are returned

Returns:
An object of the class

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 43 of file BatchProcessing.py.

```
    def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject):

        """
    The __init__ function is the constructor for a class. It is called when an object of that class
    is created, and it sets up the attributes of that object. In this case, we are setting up our
    object to have a dataframe attribute (which will be used to store all of our data), as well as
    attributes for each parameter in our ReST call.

    Args:
        self: Represent the instance of the class
        RestDomain: Specify the domain of the rest api
        NumBatches: Determine how many batches of data to retrieve
        ParameterDict: Pass in the parameters that will be used to make the api call
        HeaderDict: Pass the header dictionary from the main function to this class
        ColumnSelection: Determine which columns to pull from the api
        valueObject: Pass in the value object that is used to determine what values are returned

    Returns:
        An object of the class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterDict = ParameterDict
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.__columnSelection = ColumnSelection
        self.valueObject = valueObject
        self.__maxRequests = 10000
        self.__requestCount = math.ceil(self.__numBatches / (self.__maxRequests /
  int(self.__parameterDict['size'])))
        self.__requestCalls = math.ceil(self.__maxRequests / int(self.__parameterDict['size']))
        self.__dateTracker = None

```

3.2.3 Member Function Documentation

3.2.3.1 ConstructionMonitorProcessor()

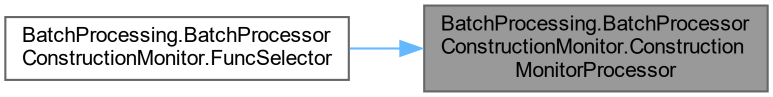
```
def BatchProcessing.BatchProcessorConstructionMonitor.ConstructionMonitorProcessor (
              self,
              valueObject )
```

```
The ConstructionMonitorProcessor function will use requests to get data from
ConstructionMontior.com's ReST API and store it into a pandas DataFrame object called __df (which is local). This
process will be repeated until all the data has been collected from ConstructionMonitor.com's ReST API, at which

Args:
self: Represent the instance of the object itself
valueObject: Update the progress bar in the gui

Returns:
A dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 94 of file BatchProcessing.py.

```
    def ConstructionMonitorProcessor(self, valueObject):
        """
    The ConstructionMonitorProcessor function will use requests to get data from
       ConstructionMontior.com's ReST API and store it into a pandas DataFrame object called __df (which
 is local). This
       process will be repeated until all the data has been collected from ConstructionMonitor.com's ReST
 API, at which point __df will contain all

    Args:
        self: Represent the instance of the object itself
        valueObject: Update the progress bar in the gui

    Returns:
        A dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __df = None
        for callNum in range(0, self.__requestCount):
            self.__parameterDict["from"] = 0

            if self.__requestCount > 1 and callNum != self.__requestCount - 1:
                __batchNum = self.__requestCalls
                if __df is None:
                    self.__dateTracker = str(date.today())
                else:
                    self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
            elif self.__requestCount == 1:
                __batchNum = self.__numBatches
                self.__dateTracker = str(date.today())
            else:
                __batchNum = self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size']))
 - (
                        self.__requestCount - 1)
                self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')

            self.__parameterDict['dateEnd'] = self.__dateTracker

            for record in range(0, int(math.ceil(__batchNum))):
                if record != 0:
                    self.__parameterDict["from"] = record * int(self.__parameterDict["size"])

                response = requests.post(url=self.__restDomain,
                                         headers=self.__headerDict,
                                         json=self.__parameterDict)

                counter = 0
                try:
                    response = response.json()['hits']['hits']
                except KeyError as e:
                    # Logging
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProcessing.py |Error = {e} | Count Request Error Server Response: {response.json()} | Batch = {record} | Parameters = {self.__parameterDict} | Headers = {self.__headerDict}")
                    continue

                valueObject.setValue(valueObject.getValue() + 1)

                if record == 0 and callNum == 0:
                    __df = pd.json_normalize(response[counter]["_source"])
                    __df["id"] = response[counter]['_id']
                    __df["county"] = response[counter]["_source"]['county']['county_name']
                    counter += 1

                for i in range(counter, len(response)):
                    __tdf = pd.json_normalize(response[i]["_source"])
                    __tdf["id"] = response[i]['_id']
                    __tdf["county"] = response[i]["_source"]['county']['county_name']
                    __df = pd.concat([__df, __tdf], ignore_index=True)

        if self.__columnSelection is not None:
            __col_list = StringToList(self.__columnSelection)
            __col_list.append("id")
            __col_list.append("county")
        else:
            pass

        self.dataframe = __df
        valueObject.setValue(-999)


```

Here is the caller graph for this function:


BatchProcessing.BatchProcessor ConstructionMonitor.FuncSelector
BatchProcessing.BatchProcessor ConstructionMonitor.Construction MonitorProcessor

3.2.3.2 FuncSelector()

```
def BatchProcessing.BatchProcessorConstructionMonitor.FuncSelector (
              self )
```

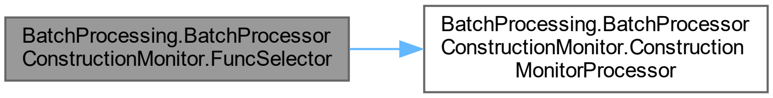
```
The FuncSelector function is a function that takes the valueObject and passes it to the ConstructionMonitorProces
The ConstructionMonitorProcessor function then uses this valueObject to determine which of its functions should b

Args:
```

```
self: Represent the instance of the class

Returns:
The result of the constructionmonitorprocessor function

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 78 of file BatchProcessing.py.

```
    def FuncSelector(self):
        """
    The FuncSelector function is a function that takes the valueObject and passes it to the
ConstructionMonitorProcessor function.
    The ConstructionMonitorProcessor function then uses this valueObject to determine which of its
functions should be called.

    Args:
        self: Represent the instance of the class

    Returns:
        The result of the constructionmonitorprocessor function

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.ConstructionMonitorProcessor(self.valueObject)
```

Here is the call graph for this function:


BatchProcessing.BatchProcessor ConstructionMonitor.FuncSelector
BatchProcessing.BatchProcessor ConstructionMonitor.Construction MonitorProcessor

3.2.4 Member Data Documentation

3.2.4.1 __columnSelection

`BatchProcessing.BatchProcessorConstructionMonitor.__columnSelection [private]`

Definition at line 71 of file BatchProcessing.py.

3.2.4.2 __dateTracker

`BatchProcessing.BatchProcessorConstructionMonitor.__dateTracker [private]`

Definition at line 76 of file BatchProcessing.py.

3.2.4.3 __headerDict

`BatchProcessing.BatchProcessorConstructionMonitor.__headerDict [private]`

Definition at line 70 of file BatchProcessing.py.

3.2.4.4 __maxRequests

`BatchProcessing.BatchProcessorConstructionMonitor.__maxRequests [private]`

Definition at line 73 of file BatchProcessing.py.

3.2.4.5 __numBatches

`BatchProcessing.BatchProcessorConstructionMonitor.__numBatches [private]`

Definition at line 67 of file BatchProcessing.py.

3.2.4.6 __parameterDict

`BatchProcessing.BatchProcessorConstructionMonitor.__parameterDict [private]`

Definition at line 68 of file BatchProcessing.py.

3.2.4.7 __requestCalls

`BatchProcessing.BatchProcessorConstructionMonitor.__requestCalls [private]`

Definition at line 75 of file BatchProcessing.py.

3.2.4.8 __requestCount

`BatchProcessing.BatchProcessorConstructionMonitor.__requestCount [private]`

Definition at line 74 of file BatchProcessing.py.

3.2.4.9 __restDomain

```
BatchProcessing.BatchProcessorConstructionMonitor.__restDomain  [private]
```

Definition at line 69 of file BatchProcessing.py.

3.2.4.10 dataframe

```
BatchProcessing.BatchProcessorConstructionMonitor.dataframe
```

Definition at line 66 of file BatchProcessing.py.

3.2.4.11 valueObject

```
BatchProcessing.BatchProcessorConstructionMonitor.valueObject
```

Definition at line 72 of file BatchProcessing.py.

The documentation for this class was generated from the following file:

- BatchProcessing.py

3.3 BatchProcessing.BatchProcessorUtahRealEstate Class Reference

Public Member Functions

- def __init__ (self, RestDomain, NumBatches, ParameterString, HeaderDict, valueObject)
- def FuncSelector (self)
- def BatchProcessingUtahRealestateCom (self, valueObject)

Public Attributes

- dataframe
- valueObject

Private Attributes

- __numBatches
- __parameterString
- __restDomain
- __headerDict

3.3.1 Detailed Description

Definition at line 172 of file BatchProcessing.py.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 __init__()

```
def BatchProcessing.BatchProcessorUtahRealEstate.__init__ (
              self,
              RestDomain,
              NumBatches,
              ParameterString,
              HeaderDict,
              valueObject )
```

```
The __init__ function is the constructor for a class. It is called when an object of that class
is instantiated, and it sets up the attributes of that object. In this case, we are setting up
the dataframe attribute to be None (which will be set later), and we are also setting up some
other attributes which will help us make our API calls.

Args:
self: Represent the instance of the class
RestDomain: Specify the domain of the rest api
NumBatches: Determine how many batches of data to pull from the api
ParameterString: Pass the parameters to the rest api
HeaderDict: Pass in the header information for the api call
valueObject: Create a dataframe from the json response

Returns:
The instance of the class

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 174 of file BatchProcessing.py.

```
    def __init__(self, RestDomain, NumBatches, ParameterString, HeaderDict, valueObject):
        """
    The __init__ function is the constructor for a class. It is called when an object of that class
    is instantiated, and it sets up the attributes of that object. In this case, we are setting up
    the dataframe attribute to be None (which will be set later), and we are also setting up some
    other attributes which will help us make our API calls.

    Args:
        self: Represent the instance of the class
        RestDomain: Specify the domain of the rest api
        NumBatches: Determine how many batches of data to pull from the api
        ParameterString: Pass the parameters to the rest api
        HeaderDict: Pass in the header information for the api call
        valueObject: Create a dataframe from the json response

    Returns:
        The instance of the class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterString = ParameterString
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.valueObject = valueObject

```

3.3.3 Member Function Documentation

3.3.3.1 BatchProcessingUtahRealestateCom()

```
def BatchProcessing.BatchProcessorUtahRealEstate.BatchProcessingUtahRealestateCom (
              self,
              valueObject )
```

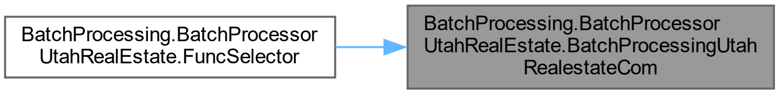
```
The BatchProcessingUtahRealestateCom function is a function that takes in the valueObject and uses it to
update the progress bar. It also takes in self, which contains all the necessary information for this
function to work properly. The BatchProcessingUtahRealestateCom function will then use requests to get data from
UtahRealestate.com's ReST API and store it into a pandas DataFrame object called __df (which is local). This
process will be repeated until all the data has been collected from UtahRealestate.com's ReST API, at which point

Args:
self: Represent the instance of the class
valueObject: Pass the value of a progress bar to the function

Returns:
A dataframe of the scraped data

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 219 of file BatchProcessing.py.

```
00219     def BatchProcessingUtahRealestateCom(self, valueObject):
00220         """
00221     The BatchProcessingUtahRealestateCom function is a function that takes in the valueObject and uses it
      to
00222        update the progress bar. It also takes in self, which contains all the necessary information for
      this
00223        function to work properly. The BatchProcessingUtahRealestateCom function will then use requests to
      get data from
00224        UtahRealestate.com's ReST API and store it into a pandas DataFrame object called __df (which is
      local). This
00225        process will be repeated until all the data has been collected from UtahRealestate.com's ReST API,
      at which point __df will contain all
00226
00227     Args:
00228         self: Represent the instance of the class
00229         valueObject: Pass the value of a progress bar to the function
00230
00231     Returns:
00232         A dataframe of the scraped data
00233
00234     Doc Author:
00235         Willem van der Schans, Trelent AI
00236     """
00237         __df = pd.DataFrame()
00238
00239         for batch in range(self.__numBatches):
00240
00241             if batch == 0:
00242                 response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
00243                                         headers=self.__headerDict)
00244
00245                 response_temp = response.json()
00246                 __df = pd.json_normalize(response_temp, record_path=['value'])
00247
00248             else:
00249                 response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch
      * 200}",
00250                                         headers=self.__headerDict)
00251
00252                 response_temp = response.json()
```

```
                response_temp = pd.json_normalize(response_temp, record_path=['value'])
                __df = pd.concat([__df, response_temp], ignore_index=True)

            valueObject.setValue(valueObject.getValue() + 1)

        self.dataframe = __df
        valueObject.setValue(-999)
```

Here is the caller graph for this function:


BatchProcessing.BatchProcessor UtahRealEstate.FuncSelector
BatchProcessing.BatchProcessor UtahRealEstate.BatchProcessingUtah RealestateCom

3.3.3.2 FuncSelector()

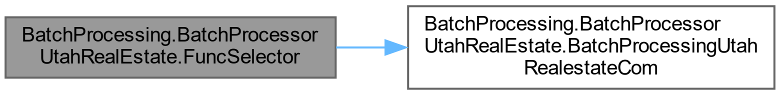
```
def BatchProcessing.BatchProcessorUtahRealEstate.FuncSelector (
              self )
```

```
The FuncSelector function is a function that takes the valueObject as an argument and then calls the appropriate
function based on what was selected in the dropdown menu.  The valueObject is passed to each of these functions
so that they can access all of its attributes.

Args:
self: Represent the instance of the class

Returns:
The function that is selected by the user

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 202 of file BatchProcessing.py.

```
    def FuncSelector(self):
        """
    The FuncSelector function is a function that takes the valueObject as an argument and then calls the appropriate
        function based on what was selected in the dropdown menu.  The valueObject is passed to each of these functions
        so that they can access all of its attributes.

    Args:
        self: Represent the instance of the class

    Returns:
        The function that is selected by the user

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.BatchProcessingUtahRealestateCom(self.valueObject)

```

Here is the call graph for this function:


BatchProcessing.BatchProcessor UtahRealEstate.FuncSelector
BatchProcessing.BatchProcessor UtahRealEstate.BatchProcessingUtah RealestateCom

3.3.4 Member Data Documentation

3.3.4.1 __headerDict

`BatchProcessing.BatchProcessorUtahRealEstate.__headerDict [private]`

Definition at line 199 of file BatchProcessing.py.

3.3.4.2 __numBatches

`BatchProcessing.BatchProcessorUtahRealEstate.__numBatches [private]`

Definition at line 196 of file BatchProcessing.py.

3.3.4.3 __parameterString

`BatchProcessing.BatchProcessorUtahRealEstate.__parameterString [private]`

Definition at line 197 of file BatchProcessing.py.

3.3.4.4 __restDomain

`BatchProcessing.BatchProcessorUtahRealEstate.__restDomain [private]`

Definition at line 198 of file BatchProcessing.py.

3.3.4.5 dataframe

```
BatchProcessing.BatchProcessorUtahRealEstate.dataframe
```

Definition at line 195 of file BatchProcessing.py.

3.3.4.6 valueObject

```
BatchProcessing.BatchProcessorUtahRealEstate.valueObject
```

Definition at line 200 of file BatchProcessing.py.

The documentation for this class was generated from the following file:

- BatchProcessing.py

3.4 BatchProgressGUI.BatchProgressGUI Class Reference

Public Member Functions

- def __init__ (self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None)
- def BatchGuiShow (self)
- def CreateProgressLayout (self)
- def createGui (self, Sourcetype)
- def ProgressUpdater (self, valueObj)
- def TimeUpdater (self, start_time)
- def ValueChecker (self, ObjectVal)

Public Attributes

- dataframe

Private Attributes

- __parameterDict
- __restDomain
- __headerDict
- __columnSelection
- __type
- __layout
- __batches
- __window
- __batch_counter

3.4.1 Detailed Description

Definition at line 17 of file BatchProgressGUI.py.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 __init__()

```
def BatchProgressGUI.BatchProgressGUI.__init__ (
              self,
              BatchesNum,
              RestDomain,
              ParameterDict,
              HeaderDict,
              Type,
              ColumnSelection = None )
```

```
The __init__ function is the first function that gets called when an object of this class is created.
It initializes all the variables and sets up a layout for the GUI. It also creates a window to display
the dataframe in.

Args:
self: Represent the instance of the class
BatchesNum: Determine the number of batches that will be created
RestDomain: Specify the domain of the rest api
ParameterDict: Pass the parameters of the request to the class
HeaderDict: Store the headers of the dataframe
Type: Determine the type of dataframe that is being created
ColumnSelection: Select the columns to be displayed in the gui

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 19 of file BatchProgressGUI.py.

```
    def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None):

        """
    The __init__ function is the first function that gets called when an object of this class is created.
    It initializes all the variables and sets up a layout for the GUI. It also creates a window to display
    the dataframe in.

    Args:
        self: Represent the instance of the class
        BatchesNum: Determine the number of batches that will be created
        RestDomain: Specify the domain of the rest api
        ParameterDict: Pass the parameters of the request to the class
        HeaderDict: Store the headers of the dataframe
        Type: Determine the type of dataframe that is being created
        ColumnSelection: Select the columns to be displayed in the gui

    Returns:
        Nothing

    Doc Author:
```

```
00039         Willem van der Schans, Trelent AI
00040     """
00041         self.__parameterDict = ParameterDict
00042         self.__restDomain = RestDomain
00043         self.__headerDict = HeaderDict
00044         self.__columnSelection = ColumnSelection
00045         self.__type = Type
00046         self.dataframe = None
00047 
00048         self.__layout = None
00049         self.__batches = BatchesNum
00050         self.__window = None
00051         self.__batch_counter = 0
00052 
```

3.4.3 Member Function Documentation

3.4.3.1 BatchGuiShow()

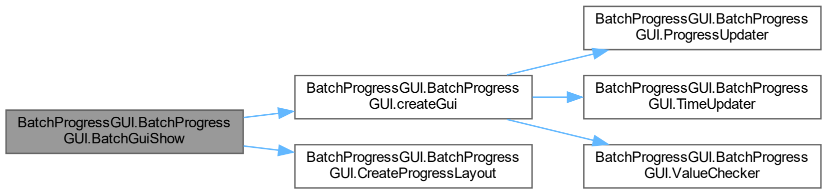
```
def BatchProgressGUI.BatchProgressGUI.BatchGuiShow (
              self )
```

```
The BatchGuiShow function is called by the BatchGui function. It creates a progress bar layout and then calls the

Args:
self: Represent the instance of the class

Returns:
The __type of the batchgui class

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 53 of file BatchProgressGUI.py.

```
00053     def BatchGuiShow(self):
00054         """
00055     The BatchGuiShow function is called by the BatchGui function. It creates a progress bar layout and
      then calls the createGui function to create a GUI for batch processing.
00056 
00057     Args:
00058         self: Represent the instance of the class
00059 
00060     Returns:
00061         The __type of the batchgui class
00062 
00063     Doc Author:
00064         Willem van der Schans, Trelent AI
00065     """
00066         self.CreateProgressLayout()
00067         self.createGui(self.__type)
00068 
```

Here is the call graph for this function:


BatchProgressGUI.BatchProgressGUI.BatchGuiShow
BatchProgressGUI.BatchProgressGUI.createGui
BatchProgressGUI.BatchProgressGUI.CreateProgressLayout
BatchProgressGUI.BatchProgressGUI.ProgressUpdater
BatchProgressGUI.BatchProgressGUI.TimeUpdater
BatchProgressGUI.BatchProgressGUI.ValueChecker

3.4.3.2 createGui()

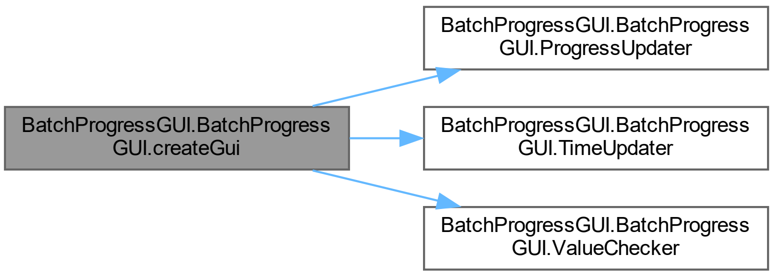
```
def BatchProgressGUI.BatchProgressGUI.createGui (
              self,
              Sourcetype )
```

```
The createGui function is the main function that creates the GUI.
It takes in a type parameter which determines what kind of batch processor to use.
The createGui function then sets up all the variables and objects needed for
the program to run, including: window, start_time, update_text, valueObj (DataTransfer),
processorObject (BatchProcessorConstructionMonitor or BatchProcessorUtahRealestate),
and threading objects for TimeUpdater and ValueChecker functions. The createGui function also starts these thread

Args:
self: Access the object itself
Sourcetype: Determine which batch processor to use

Returns:
The dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

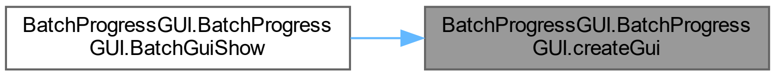
Definition at line 104 of file BatchProgressGUI.py.

```
00104     def createGui(self, Sourcetype):
00105 
00106         """
00107     The createGui function is the main function that creates the GUI.
00108     It takes in a type parameter which determines what kind of batch processor to use.
00109     The createGui function then sets up all the variables and objects needed for
00110     the program to run, including: window, start_time, update_text, valueObj (DataTransfer),
00111     processorObject (BatchProcessorConstructionMonitor or BatchProcessorUtahRealestate),
00112     and threading objects for TimeUpdater and ValueChecker functions. The createGui function also starts
      these threads.
00113 
00114     Args:
00115         self: Access the object itself
00116         Sourcetype: Determine which batch processor to use
00117 
00118     Returns:
00119         The dataframe
00120 
00121     Doc Author:
00122         Willem van der Schans, Trelent AI
00123     """
00124         self.__window = sg.Window('Progress', self.__layout, finalize=True,
      icon=ImageLoader("taskbar_icon.ico"))
00125 
00126         start_time = datetime.datetime.now().replace(microsecond=0)
00127         update_text = f"Batch {0} completed"
00128         self.__window['--progress_text--'].update(update_text)
00129         self.__window['--progress_bar--'].update(0)
00130         self.__window['--time_est--'].update("Est time needed 00:00:00")
00131 
00132         valueObj = DataTransfer()
00133         valueObj.setValue(0)
00134 
00135         if Sourcetype == "construction_monitor":
00136 
00137             processorObject = BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
00138                                                                 NumBatches=self.__batches,
00139                                                                 ParameterDict=self.__parameterDict,
00140                                                                 HeaderDict=self.__headerDict,
00141                                                                 ColumnSelection=self.__columnSelection,
00142                                                                 valueObject=valueObj)
00143         elif Sourcetype == "utah_real_estate":
00144             processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
                                                                NumBatches=self.__batches,
                                                                ParameterString=self.__parameterDict,
                                                                HeaderDict=self.__headerDict,
                                                                valueObject=valueObj)

        threading.Thread(target=self.TimeUpdater,
                         args=(start_time,),
                         daemon=True).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | TimeUpdater Thread
Successfully Started")

        batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
                                           daemon=False)
        batchFuncThread.start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchFunc Thread
Successfully Started")
        threading.Thread(target=self.ValueChecker,
                         args=(valueObj,),
                         daemon=False).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | ValueChecker Thread
Successfully Started")

        while True:

            self.ProgressUpdater(valueObj)

            if valueObj.getValue() == -999:
                break

            window, event, values = sg.read_all_windows()
            if event.startswith('update'):
                __key_to_update = event[len('update'):]
                window[__key_to_update].update(values[event])
                window.refresh()
                pass

            if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
                break

            time.sleep(0.1)

        self.dataframe = processorObject.dataframe
        self.__window.close()

        PopupWrapped(text="Api Request Completed", windowType="notice")

```

Here is the call graph for this function:


BatchProgressGUI.BatchProgress GUI.createGui
BatchProgressGUI.BatchProgress GUI.ProgressUpdater
BatchProgressGUI.BatchProgress GUI.TimeUpdater
BatchProgressGUI.BatchProgress GUI.ValueChecker

Here is the caller graph for this function:


BatchProgressGUI.BatchProgress GUI.BatchGuiShow
BatchProgressGUI.BatchProgress GUI.createGui

3.4.3.3 CreateProgressLayout()

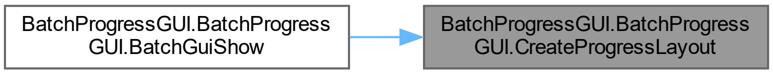
```
def BatchProgressGUI.BatchProgressGUI.CreateProgressLayout (
              self )
```

```
The CreateProgressLayout function creates the layout for the progress window.
The function takes in self as a parameter and returns nothing.

Parameters:
    self (object): The object that is calling this function.

Args:
self: Access the class variables and methods

Returns:
A list of lists

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 69 of file BatchProgressGUI.py.

```
00069     def CreateProgressLayout(self):
00070 
00071         """
00072     The CreateProgressLayout function creates the layout for the progress window.
00073         The function takes in self as a parameter and returns nothing.
00074 
00075         Parameters:
00076             self (object): The object that is calling this function.
00077 
00078     Args:
00079         self: Access the class variables and methods
00080 
00081     Returns:
00082         A list of lists
00083 
00084     Doc Author:
00085         Willem van der Schans, Trelent AI
00086     """
00087         sg.theme('Default1')
00088 
00089         __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center",
      key="--progress_text--"),
00090                    sg.Push()]
00091 
00092         __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center", key="--timer--"),
00093                    sg.Text(font=("Helvetica", 10), justification="center", key="--time_est--"), sg.Push()]
00094 
00095         __Line3 = [
            sg.ProgressBar(max_value=self.__batches, bar_color=("#920303", "#C9c8c8"), orientation='h',
    size=(30, 20),
                           key='--progress_bar--')]


        layout = [__Line1, __Line2, __Line3]

        self.__layout = layout

```

Here is the caller graph for this function:


BatchProgressGUI.BatchProgressGUI.BatchGuiShow
BatchProgressGUI.BatchProgressGUI.CreateProgressLayout

3.4.3.4 ProgressUpdater()

```
def BatchProgressGUI.BatchProgressGUI.ProgressUpdater (
              self,
              valueObj )
```

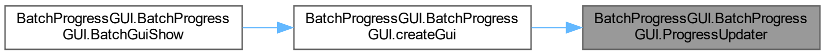
```
The ProgressUpdater function is a callback function that updates the progress bar and text
in the GUI. It takes in one argument, which is an object containing information about the
current batch number. The ProgressUpdater function then checks if this value has changed from
the last time it was called (i.e., if we are on a new batch). If so, it updates both the progress
bar and text with this new information.

Args:
self: Make the progressupdater function an instance method
valueObj: Get the current value of the batch counter

Returns:
The value of the batch counter

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 188 of file BatchProgressGUI.py.

```
    def ProgressUpdater(self, valueObj):
        """
    The ProgressUpdater function is a callback function that updates the progress bar and text
    in the GUI. It takes in one argument, which is an object containing information about the
    current batch number. The ProgressUpdater function then checks if this value has changed from
    the last time it was called (i.e., if we are on a new batch). If so, it updates both the progress
    bar and text with this new information.

    Args:
        self: Make the progressupdater function an instance method
        valueObj: Get the current value of the batch counter

    Returns:
        The value of the batch counter

```

```
00203     Doc Author:
00204         Willem van der Schans, Trelent AI
00205     """
00206         if valueObj.getValue() != self.__batch_counter:
00207             self.__batch_counter = valueObj.getValue()
00208 
00209             __update_text = f"Batch {self.__batch_counter}/{self.__batches} completed"
00210 
00211             self.__window.write_event_value('update--progress_bar--', self.__batch_counter)
00212             self.__window.write_event_value('update--progress_text--', __update_text)
00213         else:
00214             pass
00215 
```

Here is the caller graph for this function:


BatchProgressGUI.BatchProgress GUI.BatchGuiShow
BatchProgressGUI.BatchProgress GUI.createGui
BatchProgressGUI.BatchProgress GUI.ProgressUpdater

3.4.3.5 TimeUpdater()

```
def BatchProgressGUI.BatchProgressGUI.TimeUpdater (
                self,
                start_time )
```

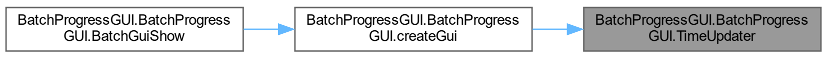
```
The TimeUpdater function is a thread that updates the time elapsed and estimated time needed to complete
the current batch. It does this by reading the start_time variable passed in, getting the current time,
calculating how much time has passed since start_time was set and then updating a timer string with that value.
It then calculates an estimation of how long it will take to finish all batches based on how many batches have bee

Args:
self: Make the function a method of the class
start_time: Get the time when the function is called

Returns:
A string that is updated every 0

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 216 of file BatchProgressGUI.py.

```
00216     def TimeUpdater(self, start_time):
00217 
00218         """
00219     The TimeUpdater function is a thread that updates the time elapsed and estimated time needed to
      complete
00220     the current batch. It does this by reading the start_time variable passed in, getting the current
      time,
00221     calculating how much time has passed since start_time was set and then updating a timer string with
      that value.
00222     It then calculates an estimation of how long it will take to finish all batches based on how many
      batches have been completed so far.
00223 
00224     Args:
00225         self: Make the function a method of the class
00226         start_time: Get the time when the function is called
```

```

    Returns:
        A string that is updated every 0

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            if self.__batch_counter < self.__batches:

                __current_time = datetime.datetime.now().replace(microsecond=0)

                __passed_time = __current_time - start_time

                __timer_string = f"Time Elapsed {__passed_time}"

                try:
                    self.__window.write_event_value('update--timer--', __timer_string)
                except AttributeError as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProgressGUI.py | Error = {e} | Timer string attribute error, this is okay if the display looks good, this exception omits fatal crashes due to an aesthetic error")
                    break

                __passed_time = __passed_time.total_seconds()

                try:
                    __time_est = datetime.timedelta(
                        seconds=(__passed_time * (self.__batches / self.__batch_counter) - __passed_time)).seconds
                except:
                    __time_est = datetime.timedelta(
                        seconds=(__passed_time * self.__batches - __passed_time)).seconds

                __time_est = time.strftime('%H:%M:%S', time.gmtime(__time_est))

                __end_string = f"Est time needed {__time_est}"
                self.__window.write_event_value('update--time_est--', __end_string)
            else:
                __end_string = f"Est time needed 00:00:00"
                self.__window.write_event_value('update--time_est--', __end_string)
            time.sleep(0.25)

```

Here is the caller graph for this function:


BatchProgressGUI.BatchProgress GUI.BatchGuiShow
BatchProgressGUI.BatchProgress GUI.createGui
BatchProgressGUI.BatchProgress GUI.TimeUpdater

3.4.3.6 ValueChecker()

```
def BatchProgressGUI.BatchProgressGUI.ValueChecker (
                self,
                ObjectVal )
```

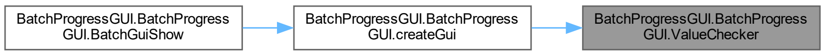
```
The ValueChecker function is a thread that checks the value of an object.
It will check if the value has changed, and if it has, it will return True.
If not, then it returns False.
```

```
Args:
self: Represent the instance of the class
ObjectVal: Get the value of the object

Returns:
True if the value of the object has changed, and false if it hasn't

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 268 of file BatchProgressGUI.py.

```
    def ValueChecker(self, ObjectVal):
        """
    The ValueChecker function is a thread that checks the value of an object.
        It will check if the value has changed, and if it has, it will return True.
        If not, then it returns False.

    Args:
        self: Represent the instance of the class
        ObjectVal: Get the value of the object

    Returns:
        True if the value of the object has changed, and false if it hasn't

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            time.sleep(0.3)
            if self.__batch_counter != ObjectVal.getValue():
                self.__batch_counter = ObjectVal.getValue()
                return True
            else:
                return False
```

Here is the caller graph for this function:


BatchProgressGUI.BatchProgress GUI.BatchGuiShow
BatchProgressGUI.BatchProgress GUI.createGui
BatchProgressGUI.BatchProgress GUI.ValueChecker

3.4.4 Member Data Documentation

3.4.4.1 __batch_counter

`BatchProgressGUI.BatchProgressGUI.__batch_counter [private]`

Definition at line 51 of file BatchProgressGUI.py.

3.4.4.2 __batches

`BatchProgressGUI.BatchProgressGUI.__batches [private]`

Definition at line 49 of file BatchProgressGUI.py.

3.4.4.3 __columnSelection

`BatchProgressGUI.BatchProgressGUI.__columnSelection [private]`

Definition at line 44 of file BatchProgressGUI.py.

3.4.4.4 __headerDict

`BatchProgressGUI.BatchProgressGUI.__headerDict [private]`

Definition at line 43 of file BatchProgressGUI.py.

3.4.4.5 __layout

`BatchProgressGUI.BatchProgressGUI.__layout [private]`

Definition at line 48 of file BatchProgressGUI.py.

3.4.4.6 __parameterDict

`BatchProgressGUI.BatchProgressGUI.__parameterDict [private]`

Definition at line 41 of file BatchProgressGUI.py.

3.4.4.7 __restDomain

`BatchProgressGUI.BatchProgressGUI.__restDomain [private]`

Definition at line 42 of file BatchProgressGUI.py.

3.4.4.8 __type

```
BatchProgressGUI.BatchProgressGUI.__type  [private]
```

Definition at line 45 of file BatchProgressGUI.py.

3.4.4.9 __window

```
BatchProgressGUI.BatchProgressGUI.__window  [private]
```

Definition at line 50 of file BatchProgressGUI.py.

3.4.4.10 dataframe

```
BatchProgressGUI.BatchProgressGUI.dataframe
```

Definition at line 46 of file BatchProgressGUI.py.

The documentation for this class was generated from the following file:

- BatchProgressGUI.py

3.5 Core.CFBP Class Reference

Public Member Functions

- def __init__ (self, state_arg=None, year_arg=None)

Public Attributes

- state_arg
- year_arg
- uiString
- link

Private Member Functions

- def __showUi (self)
- def __dataGetter (self)

3.5.1 Detailed Description

Definition at line 15 of file CFBP/Core.py.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 __init__()

```
def Core.CFBP.__init__ (
                self,
                state_arg = None,
                year_arg = None )
```

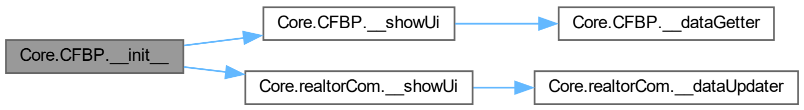
```
The __init__ function is called when the class is instantiated.
Its job is to initialize the object with some default values, and do any other setup that might be necessary.
The __init__ function can take arguments, but it doesn't have to.

Args:
self: Represent the instance of the class
state_arg: Set the state_arg attribute of the class
year_arg: Set the year of data to be retrieved

Returns:
A popupwrapped object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 17 of file CFBP/Core.py.

```
    def __init__(self, state_arg=None, year_arg=None):
        """
    The __init__ function is called when the class is instantiated.
    Its job is to initialize the object with some default values, and do any other setup that might be
necessary.
    The __init__ function can take arguments, but it doesn't have to.

    Args:
        self: Represent the instance of the class
        state_arg: Set the state_arg attribute of the class
        year_arg: Set the year of data to be retrieved

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.state_arg = state_arg
        self.year_arg = year_arg
        self.uiString = None
        self.link = None

        eventReturn = confirmDialog()
        if eventReturn == "Continue":
            startTime = datetime.datetime.now().replace(microsecond=0)
            self.__showUi()
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | API Link =
{self.link}")
            F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
```

```
00046             print(
00047                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Data retrieved with
      in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) -
      startTime).total_seconds()))}")
00048
00049             self.uiString = (
00050                 f"ffiec.cfpb.gov (Mortgage API) request Completed \n {self.year_arg} data retrieved \n
      Data Saved at {F.getPath()}")
00051
00052             PopupWrapped(text=self.uiString, windowType="noticeLarge")
00053         else:
00054             print(
00055                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled
      Request")
00056             pass
00057
```

Here is the call graph for this function:



3.5.3 Member Function Documentation

3.5.3.1 __dataGetter()

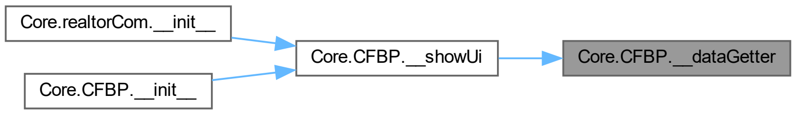
```
def Core.CFBP.__dataGetter (
              self )  [private]
```

```
The __dataGetter function is a private function that gets the data from the CFPB API.
It takes no arguments, but uses self.state_arg and self.year_arg to create a URL for the API call.

Args:
self: Represent the instance of the class

Returns:
A response object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 86 of file CFBP/Core.py.

```
00086     def __dataGetter(self):
00087         """
00088     The __dataGetter function is a private function that gets the data from the CFPB API.
00089     It takes no arguments, but uses self.state_arg and self.year_arg to create a URL for the API call.
00090
00091     Args:
00092         self: Represent the instance of the class
00093
00094     Returns:
        A response object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        arg_dict_bu = locals()

        link = settings.settingCFBPLink

        if self.state_arg is None:
            self.state_arg = "UT"
        else:
            pass

        if self.year_arg is None:
            self.year_arg = str(datetime.date.today().year - 1)
        else:
            pass

        passFlag = False

        while not passFlag:

            self.link = link + f"states={self.state_arg}" + f"&years={self.year_arg}"

            response = requests.get(self.link)

            if response.status_code == 400:
                self.year_arg = int(self.year_arg) - 1

            else:
                passFlag = True

        RESTError(response)
        raise SystemExit(0)
```

Here is the caller graph for this function:



3.5.3.2 __showUi()

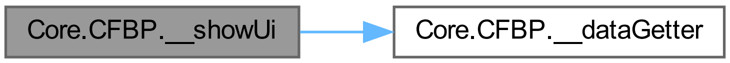
```
def Core.CFBP.__showUi (
              self ) [private]
```

```
The __showUi function is a function that creates a progress bar window.
The __showUi function takes class variables and returns a windowobj.


Args:
self: Represent the instance of the class

Returns:
The uiobj variable

Doc Author:
Willem van der Schans, Trelent AI
```

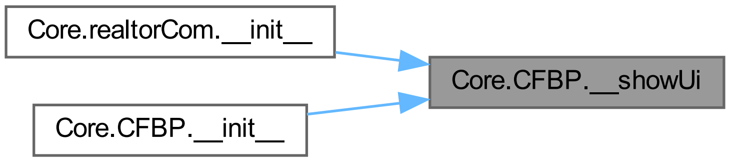
Definition at line 58 of file CFBP/Core.py.

```
    def __showUi(self):

        """
    The __showUi function is a function that creates a progress bar window.
    The __showUi function takes class variables and returns a windowobj.


    Args:
        self: Represent the instance of the class

    Returns:
        The uiobj variable

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Cenus Request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__dataGetter,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()

```

Here is the call graph for this function:



Here is the caller graph for this function:



3.5.4 Member Data Documentation

3.5.4.1 link

```
Core.CFBP.link
```

Definition at line 37 of file CFBP/Core.py.

3.5.4.2 state_arg

```
Core.CFBP.state_arg
```

Definition at line 34 of file CFBP/Core.py.

3.5.4.3 uiString

```
Core.CFBP.uiString
```

Definition at line 36 of file CFBP/Core.py.

3.5.4.4 year_arg

```
Core.CFBP.year_arg
```

Definition at line 35 of file CFBP/Core.py.

The documentation for this class was generated from the following file:

- CFBP/Core.py

3.6 Core.ConstructionMonitorInit Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- size
- SourceInclude
- dateStart
- dateEnd
- rest_domain
- auth_key
- ui_flag
- append_file

Private Member Functions

- def __ShowGui (self, layout, text)
- def __SetValues (self, values)

Static Private Member Functions

- def __CreateFrame ()

3.6.1 Detailed Description

Definition at line 25 of file ConstructionMonitor/Core.py.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 __init__()

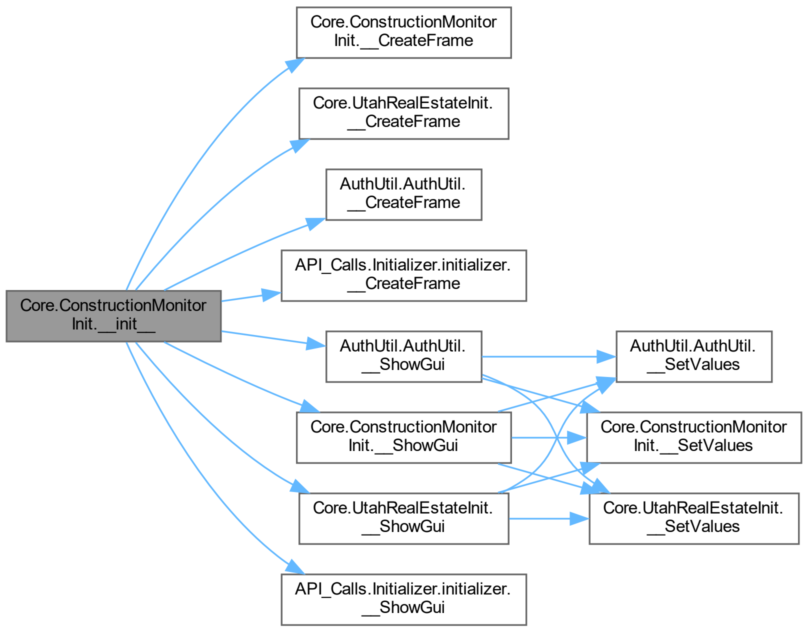
```
def Core.ConstructionMonitorInit.__init__ (
              self )
```

```
The __init__ function is called when the class is instantiated.
It sets up the variables that will be used by other functions in this class.


Args:
self: Represent the instance of the class

Returns:
None

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 27 of file ConstructionMonitor/Core.py.

```
00027     def __init__(self):
00028 
00029         """
00030     The __init__ function is called when the class is instantiated.
00031     It sets up the variables that will be used by other functions in this class.
00032 
00033 
00034     Args:
00035         self: Represent the instance of the class
00036 
00037     Returns:
00038         None
00039 
00040     Doc Author:
00041         Willem van der Schans, Trelent AI
00042     """
00043         self.size = None
00044         self.SourceInclude = None
00045         self.dateStart = None
00046         self.dateEnd = None
00047         self.rest_domain = None
00048         self.auth_key = None
00049         self.ui_flag = None
00050         self.append_file = None
00051 
00052         passFlag = False
00053 
00054         while not passFlag:
00055             if os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00056                     "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
00057                 Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00058                     "Security").joinpath("auth.json")):
00059                 try:
00060                     f = open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00061                         "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00062                     key = f.readline()
00063                     f.close()
00064                     f = open(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00065                         "Security").joinpath("auth.json"), "rb")
00066                     authDict = json.load(f)
00067                     fernet = Fernet(key)
00068                     self.auth_key = fernet.decrypt(authDict["cm"]["auth"]).decode()
00069                     passFlag = True
00070                 except Exception as e:
00071                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
      ConstructionMonitor/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
00072                     AuthUtil()
00073             else:
00074                 AuthUtil()
00075 
00076         self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")
00077 
```

Here is the call graph for this function:


Core.ConstructionMonitor Init.__init__
Core.ConstructionMonitor Init.__CreateFrame
Core.UtahRealEstateInit. __CreateFrame
AuthUtil.AuthUtil. __CreateFrame
API_Calls.Initializer.initializer. __CreateFrame
AuthUtil.AuthUtil. __ShowGui
Core.ConstructionMonitor Init.__ShowGui
Core.UtahRealEstateInit. __ShowGui
API_Calls.Initializer.initializer. __ShowGui
AuthUtil.AuthUtil. __SetValues
Core.ConstructionMonitor Init.__SetValues
Core.UtahRealEstateInit. __SetValues

3.6.3 Member Function Documentation

3.6.3.1 __CreateFrame()

```
def Core.ConstructionMonitorInit.__CreateFrame ( )  [static], [private]
```

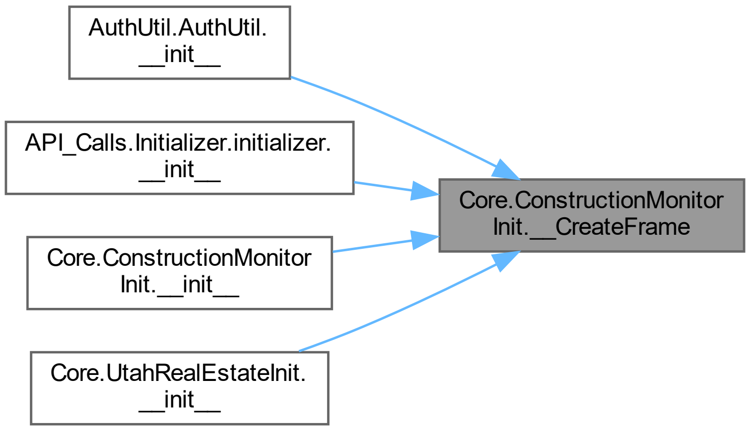
```
The __CreateFrame function creates the GUI layout for the application.
The function returns a list of lists that contains all the elements to be displayed in the GUI window.
This is done by creating each line as a list and then appending it to another list which will contain all lines.

Args:

Returns:
The layout for the gui

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 117 of file ConstructionMonitor/Core.py.

```
00117     def __CreateFrame():
00118
00119         """
00120     The __CreateFrame function creates the GUI layout for the application.
```

```
        The function returns a list of lists that contains all the elements to be displayed in the GUI window.
        This is done by creating each line as a list and then appending it to another list which will contain all lines.

    Args:

    Returns:
        The layout for the gui

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"), key="-Cal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]

        line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-EndCal-")]

        line5 = [sg.HSeparator()]

        line6 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
                          size=(20, 1)),
                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                               target="-AppendingFile-")]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout

```

Here is the caller graph for this function:



3.6.3.2 __SetValues()

```
def Core.ConstructionMonitorInit.__SetValues (
              self,
              values )  [private]
```

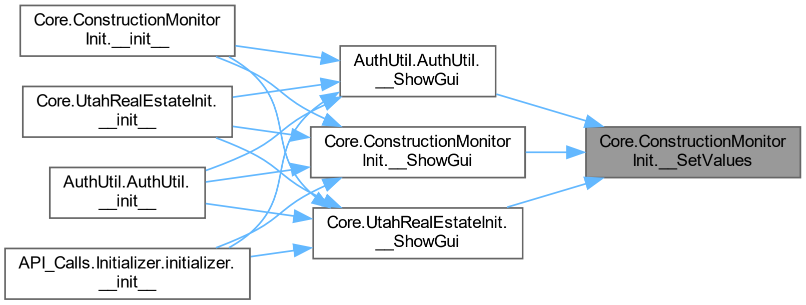
```
The __SetValues function is used to set the values of the variables that are used in the __GetData function.
The __SetValues function takes a dictionary as an argument, and then sets each variable based on what is passed i
the dictionary. The keys for this dictionary are defined by the user when they create their own instance of this

Args:
self: Represent the instance of the class
values: Pass in the values from the ui

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 176 of file ConstructionMonitor/Core.py.

```
00176     def __SetValues(self, values):
00177
00178         """
00179     The __SetValues function is used to set the values of the variables that are used in the __GetData
     function.
00180     The __SetValues function takes a dictionary as an argument, and then sets each variable based on what
     is passed into
00181     the dictionary. The keys for this dictionary are defined by the user when they create their own
     instance of this class.
00182
```

```
    Args:
        self: Represent the instance of the class
        values: Pass in the values from the ui

    Returns:
        A dictionary of values

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.size = 1000

        if values["-Cal-"] != "":
            self.dateStart = values["-Cal-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-EndCal-"] != "":
            self.dateEnd = values["-EndCal-"]
        else:
            self.dateEnd = date.today().strftime("%Y-%m-%d")

        self.rest_domain = settings.settingCMRestDomain

        self.SourceInclude = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None

        self.ui_flag = True


```

Here is the caller graph for this function:


Core.ConstructionMonitor Init.__init__
Core.UtahRealEstateInit. __init__
AuthUtil.AuthUtil. __init__
API_Calls.Initializer.initializer. __init__
AuthUtil.AuthUtil. __ShowGui
Core.ConstructionMonitor Init.__ShowGui
Core.UtahRealEstateInit. __ShowGui
Core.ConstructionMonitor Init.__SetValues

3.6.3.3 __ShowGui()

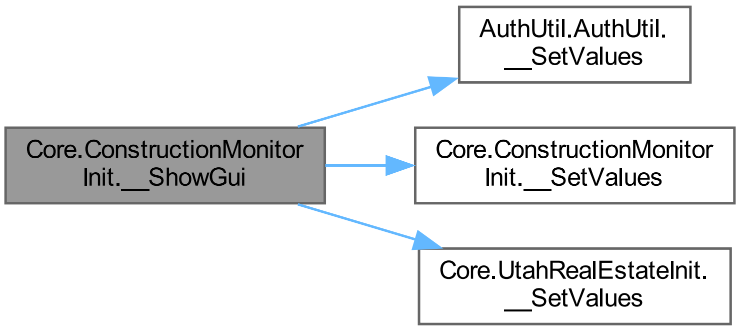
```
def Core.ConstructionMonitorInit.__ShowGui (
                self,
                layout,
                text )  [private]
```

```
The __ShowGui function is the main function that creates and displays the GUI.
It takes in a layout, which is a list of lists containing all the elements to be displayed on screen.
The text parameter specifies what title should appear at the top of the window.

Args:
self: Refer to the current instance of a class
layout: Determine what the gui will look like
text: Set the title of the window

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI
```

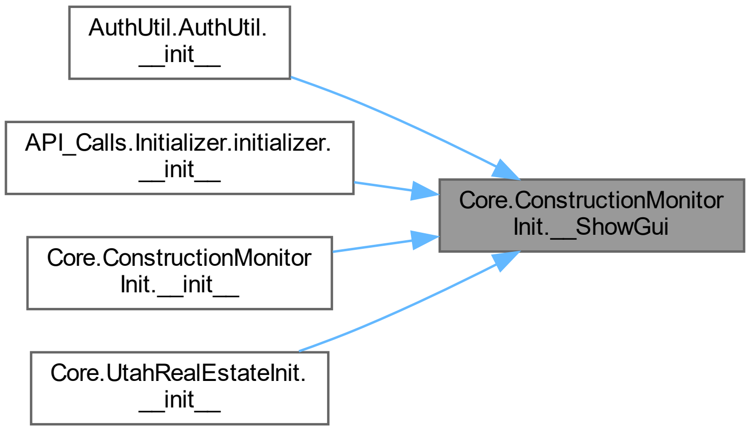
Definition at line 78 of file ConstructionMonitor/Core.py.

```
    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is the main function that creates and displays the GUI.
    It takes in a layout, which is a list of lists containing all the elements to be displayed on screen.
    The text parameter specifies what title should appear at the top of the window.

    Args:
        self: Refer to the current instance of a class
        layout: Determine what the gui will look like
        text: Set the title of the window

    Returns:
        A dictionary of values

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(933)
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

```

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.4 Member Data Documentation

3.6.4.1 append_file

`Core.ConstructionMonitorInit.append_file`

Definition at line 50 of file ConstructionMonitor/Core.py.

3.6.4.2 auth_key

```
Core.ConstructionMonitorInit.auth_key
```

Definition at line 48 of file ConstructionMonitor/Core.py.

3.6.4.3 dateEnd

```
Core.ConstructionMonitorInit.dateEnd
```

Definition at line 46 of file ConstructionMonitor/Core.py.

3.6.4.4 dateStart

```
Core.ConstructionMonitorInit.dateStart
```

Definition at line 45 of file ConstructionMonitor/Core.py.

3.6.4.5 rest_domain

```
Core.ConstructionMonitorInit.rest_domain
```

Definition at line 47 of file ConstructionMonitor/Core.py.

3.6.4.6 size

```
Core.ConstructionMonitorInit.size
```

Definition at line 43 of file ConstructionMonitor/Core.py.

3.6.4.7 SourceInclude

```
Core.ConstructionMonitorInit.SourceInclude
```

Definition at line 44 of file ConstructionMonitor/Core.py.

3.6.4.8 ui_flag

`Core.ConstructionMonitorInit.ui_flag`

Definition at line 49 of file ConstructionMonitor/Core.py.

The documentation for this class was generated from the following file:

- ConstructionMonitor/Core.py

3.7 Core.ConstructionMonitorMain Class Reference

Public Member Functions

- def __init__ (self, siteClass)
- def mainFunc (self)

Public Attributes

- dataframe

Private Member Functions

- def __ParameterCreator (self)
- def __getCount (self)
- def __getCountUI (self)

Private Attributes

- __siteClass
- __restDomain
- __headerDict
- __columnSelection
- __appendFile
- __parameterDict
- __search_id
- __record_val
- __batches
- __ui_flag

3.7.1 Detailed Description

Definition at line 217 of file ConstructionMonitor/Core.py.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 __init__()

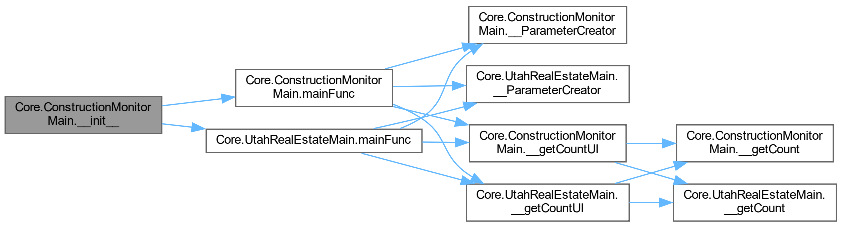
```
def Core.ConstructionMonitorMain.__init__ (
              self,
              siteClass )
```

```
The __init__ function is the first function that runs when an object of this class is created.
It sets up all the variables and functions needed for this class to run properly.


Args:
self: Represent the instance of the class
siteClass: Identify the site that is being used

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 219 of file ConstructionMonitor/Core.py.

```
    def __init__(self, siteClass):

        """
    The __init__ function is the first function that runs when an object of this class is created.
    It sets up all the variables and functions needed for this class to run properly.


    Args:
        self: Represent the instance of the class
        siteClass: Identify the site that is being used

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__siteClass = siteClass
        self.__restDomain = None
        self.__headerDict = None
        self.__columnSelection = None
        self.__appendFile = None

        self.__parameterDict = {}
        self.__search_id = None
        self.__record_val = 0
        self.__batches = 0

        self.__ui_flag = None

        self.dataframe = None

        try:
            self.mainFunc()
        except SystemError as e:
            if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Cooerced SystemError in
  ConstructionMonitorMain class")
                pass
        except AttributeError as e:
            # This allows for user cancellation of the program using the quit button
            if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Error {e}")
                pass
            elif e is not None:
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Authentication Error | Please update
      keys in AuthUtil")
                RESTError(401)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

```

Here is the call graph for this function:


Core.ConstructionMonitor Main.__init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__ParameterCreator
Core.UtahRealEstateMain. __ParameterCreator
Core.ConstructionMonitor Main.__getCountUI
Core.UtahRealEstateMain. __getCountUI
Core.ConstructionMonitor Main.__getCount
Core.UtahRealEstateMain. __getCount

3.7.3 Member Function Documentation

3.7.3.1 __getCount()

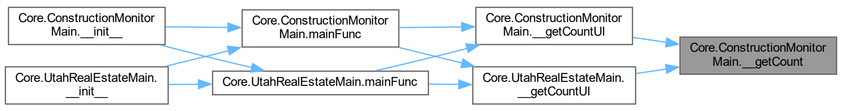
```
def Core.ConstructionMonitorMain.__getCount (
              self )  [private]
```

```
The __getCount function is used to get the total number of records that are returned from a query.
This function is called by the __init__ function and sets the self.__record_val variable with this value.

Args:
self: Represent the instance of the class

Returns:
The total number of records in the database

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 372 of file ConstructionMonitor/Core.py.

```
00372     def __getCount(self):
00373         """
00374     The __getCount function is used to get the total number of records that are returned from a query.
00375     This function is called by the __init__ function and sets the self.__record_val variable with this
      value.
00376 
00377     Args:
00378         self: Represent the instance of the class
00379 
00380     Returns:
00381         The total number of records in the database
00382 
00383     Doc Author:
00384         Willem van der Schans, Trelent AI
00385     """
00386         __count_resp = None
00387 
00388         try:
00389 
00390             __temp_param_dict = copy.copy(self.__parameterDict)
00391 
00392             __count_resp = requests.post(url=self.__restDomain,
00393                                          headers=self.__headerDict,
00394                                          json=__temp_param_dict)
00395 
00396         except requests.exceptions.Timeout as e:
00397             print(e)
00398             RESTError(790)
00399             raise SystemExit(790)
00400         except requests.exceptions.TooManyRedirects as e:
00401             print(e)
00402             RESTError(791)
00403             raise SystemExit(791)
00404         except requests.exceptions.MissingSchema as e:
00405             print(e)
00406             RESTError(1101)
00407         except requests.exceptions.RequestException as e:
00408             print(e)
00409             RESTError(405)
00410             raise SystemExit(405)
00411 
00412         __count_resp = __count_resp.json()
00413 
00414         self.__record_val = __count_resp["hits"]["total"]["value"]
00415 
00416         del __count_resp, __temp_param_dict
00417 
```

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__getCountUI
Core.UtahRealEstateMain. __getCountUI
Core.ConstructionMonitor Main.__getCount

3.7.3.2 __getCountUI()

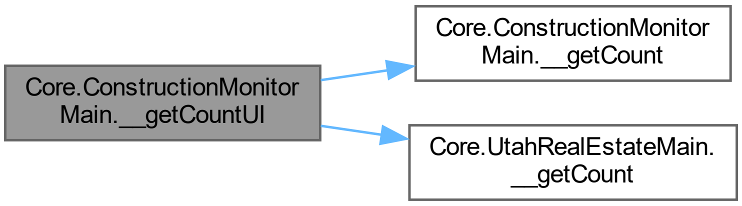
```
def Core.ConstructionMonitorMain.__getCountUI (
              self )  [private]
```

```
The __getCountUI function is a wrapper for the __getCount function.
It allows the user to run __getCount in a separate thread, so that they can continue working while it runs.
The function will display a progress bar and update with text as it progresses through its tasks.

Args:
self: Access the class variables and methods

Returns:
The count of the number of records in the database

Doc Author:
Willem van der Schans, Trelent AI
```

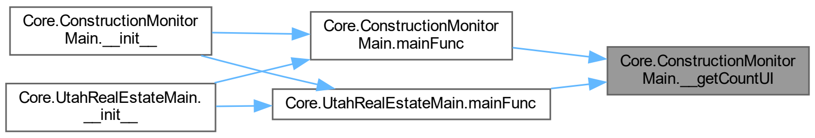
Definition at line 418 of file ConstructionMonitor/Core.py.

```
    def __getCountUI(self):

        """
    The __getCountUI function is a wrapper for the __getCount function.
    It allows the user to run __getCount in a separate thread, so that they can continue working while it
    runs.
    The function will display a progress bar and update with text as it progresses through its tasks.

    Args:
        self: Access the class variables and methods

    Returns:
        The count of the number of records in the database

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        if self.__ui_flag:
            uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

            threadGui = threading.Thread(target=self.__getCount,
                                         daemon=False)
            threadGui.start()

            while threadGui.is_alive():
                uiObj.textUpdate()
                uiObj.windowPush()
            else:
                uiObj.stopWindow()

        else:
            self.__getCount()
```

Here is the call graph for this function:


Core.ConstructionMonitorMain.__getCountUI
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCount

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__getCountUI

3.7.3.3 __ParameterCreator()

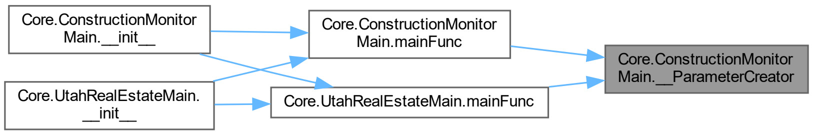
```
def Core.ConstructionMonitorMain.__ParameterCreator (
              self )  [private]
```

```
The __ParameterCreator function is used to create the parameter dictionary that will be passed into the
__Request function. The function takes in a siteClass object and extracts all of its attributes, except for
those that start with '__' or are callable. It then creates a dictionary from these attributes and stores it as
self.__parameterDict.

Args:
self: Make the function a method of the class

Returns:
A dictionary of parameters and a list of non parameter variables

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 333 of file ConstructionMonitor/Core.py.

```
    def __ParameterCreator(self):
        """
    The __ParameterCreator function is used to create the parameter dictionary that will be passed into
the
        __Request function. The function takes in a siteClass object and extracts all of its attributes,
except for
        those that start with '__' or are callable. It then creates a dictionary from these attributes and
stores it as
        self.__parameterDict.

    Args:
        self: Make the function a method of the class

    Returns:
        A dictionary of parameters and a list of non parameter variables

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                         not key.startswith('__') and not callable(key)}

        self.__restDomain = __Source_dict["rest_domain"]
        __Source_dict.pop("rest_domain")
        self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
        __Source_dict.pop("auth_key")
        self.__columnSelection = __Source_dict["SourceInclude"]
        __Source_dict.pop("SourceInclude")
        self.__ui_flag = __Source_dict["ui_flag"]
        __Source_dict.pop("ui_flag")
        self.__appendFile = __Source_dict["append_file"]
        __Source_dict.pop("append_file")

        temp_dict = copy.copy(__Source_dict)
        for key, value in temp_dict.items():
            if value is None:
                __Source_dict.pop(key)
            else:
                pass

        self.__parameterDict = copy.copy(__Source_dict)

```

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitor Main.__ParameterCreator

3.7.3.4 mainFunc()

```
def Core.ConstructionMonitorMain.mainFunc (
            self )
```

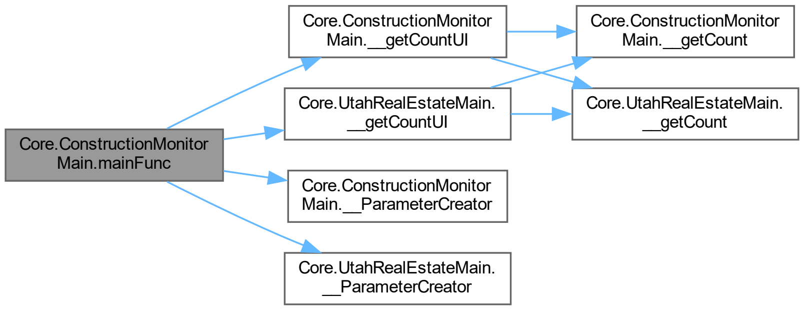
```
The mainFunc function is the main function of this module. It will be called by the GUI or CLI to execute
the code in this module. The mainFunc function will first create a parameter dictionary using the __ParameterCrea
method, then it will get a count of all records that match its parameters using the __getCountUI method, and then
it will calculate how many batches are needed to retrieve all records with those parameters using BatchCalculator
After that it asks if you want to continue with retrieving data from Salesforce (if running in GUI mode). Then it
a progress bar for each

Args:
self: Refer to the current object

Returns:
The dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 277 of file ConstructionMonitor/Core.py.

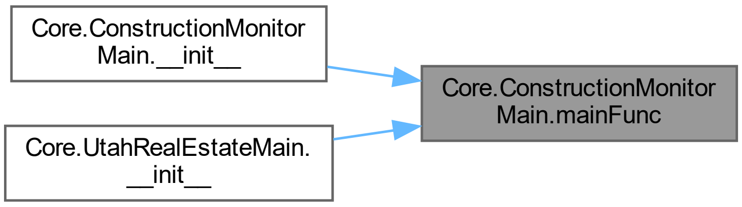
```
    def mainFunc(self):
        """
    The mainFunc function is the main function of this module. It will be called by the GUI or CLI to
execute
    the code in this module. The mainFunc function will first create a parameter dictionary using the
__ParameterCreator
    method, then it will get a count of all records that match its parameters using the __getCountUI
method, and then
    it will calculate how many batches are needed to retrieve all records with those parameters using
BatchCalculator.
```

```
    After that it asks if you want to continue with retrieving data from Salesforce (if running in GUI
mode). Then it shows
    a progress bar for each

    Args:
        self: Refer to the current object

    Returns:
        The dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__ParameterCreator()

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param Dict =
{self.__parameterDict}")
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain =
{self.__restDomain}")

        self.__getCountUI()

        self.__batches = BatchCalculator(self.__record_val, self.__parameterDict)

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches =
{self.__batches} | Rows {self.__record_val}")

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            eventReturn = BatchInputGui(self.__batches, self.__record_val)
            if eventReturn == "Continue":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for
{self.__batches} batches sent to server")
                BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                                  ParameterDict=self.__parameterDict,
                                                  HeaderDict=self.__headerDict,
                                                  ColumnSelection=self.__columnSelection,
                                                  BatchesNum=self.__batches,
                                                  Type="construction_monitor")
                BatchGuiObject.BatchGuiShow()
                self.dataframe = BatchGuiObject.dataframe
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Dataframe
retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in
{time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
                FileSaver("cm", self.dataframe, self.__appendFile)
            else:
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for
{self.__batches} batches canceled by user")
        else:
            RESTError(994)
            raise SystemExit(994)

```

Here is the call graph for this function:


Core.ConstructionMonitor Main.mainFunc
Core.ConstructionMonitor Main.__getCountUI
Core.UtahRealEstateMain. __getCountUI
Core.ConstructionMonitor Main.__ParameterCreator
Core.UtahRealEstateMain. __ParameterCreator
Core.ConstructionMonitor Main.__getCount
Core.UtahRealEstateMain. __getCount

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.ConstructionMonitor Main.mainFunc

3.7.4 Member Data Documentation

3.7.4.1 __appendFile

```
Core.ConstructionMonitorMain.__appendFile  [private]
```

Definition at line 240 of file ConstructionMonitor/Core.py.

3.7.4.2 __batches

`Core.ConstructionMonitorMain.__batches  [private]`

Definition at line 245 of file ConstructionMonitor/Core.py.

3.7.4.3 __columnSelection

`Core.ConstructionMonitorMain.__columnSelection  [private]`

Definition at line 239 of file ConstructionMonitor/Core.py.

3.7.4.4 __headerDict

`Core.ConstructionMonitorMain.__headerDict  [private]`

Definition at line 238 of file ConstructionMonitor/Core.py.

3.7.4.5 __parameterDict

`Core.ConstructionMonitorMain.__parameterDict  [private]`

Definition at line 242 of file ConstructionMonitor/Core.py.

3.7.4.6 __record_val

`Core.ConstructionMonitorMain.__record_val  [private]`

Definition at line 244 of file ConstructionMonitor/Core.py.

3.7.4.7 __restDomain

`Core.ConstructionMonitorMain.__restDomain  [private]`

Definition at line 237 of file ConstructionMonitor/Core.py.

3.7.4.8 __search_id

```
Core.ConstructionMonitorMain.__search_id [private]
```

Definition at line 243 of file ConstructionMonitor/Core.py.

3.7.4.9 __siteClass

```
Core.ConstructionMonitorMain.__siteClass [private]
```

Definition at line 236 of file ConstructionMonitor/Core.py.

3.7.4.10 __ui_flag

```
Core.ConstructionMonitorMain.__ui_flag [private]
```

Definition at line 247 of file ConstructionMonitor/Core.py.

3.7.4.11 dataframe

```
Core.ConstructionMonitorMain.dataframe
```

Definition at line 249 of file ConstructionMonitor/Core.py.

The documentation for this class was generated from the following file:

- ConstructionMonitor/Core.py

3.8 DataTransfer.DataTransfer Class Reference

Public Member Functions

- def __init__ (self)
- def setValue (self, value)
- def getValue (self)
- def whileValue (self)

Private Attributes

- __value

3.8.1 Detailed Description

Definition at line 4 of file DataTransfer.py.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 __init__()

```
def DataTransfer.DataTransfer.__init__ (
              self )
```

```
The __init__ function is called when the class is instantiated.
It sets the initial value of self.__value to 0.

Args:
self: Represent the instance of the class

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 6 of file DataTransfer.py.

```
    def __init__(self):
        """
    The __init__ function is called when the class is instantiated.
    It sets the initial value of self.__value to 0.

    Args:
        self: Represent the instance of the class

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__value = 0

```

3.8.3 Member Function Documentation

3.8.3.1 getValue()

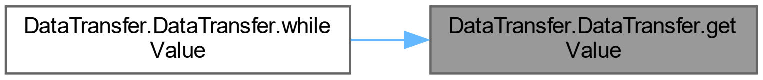
```
def DataTransfer.DataTransfer.getValue (
              self )
```

```
The getValue function returns the value of the private variable __value.
This is a getter function that allows access to this private variable.

Args:
self: Represent the instance of the class

Returns:
The value of the instance variable

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 39 of file DataTransfer.py.

```
    def getValue(self):
        """
    The getValue function returns the value of the private variable __value.
    This is a getter function that allows access to this private variable.

    Args:
        self: Represent the instance of the class

    Returns:
        The value of the instance variable

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        return self.__value

```

Here is the caller graph for this function:


DataTransfer.DataTransfer.while Value
DataTransfer.DataTransfer.get Value

3.8.3.2 setValue()

```
def DataTransfer.DataTransfer.setValue (
              self,
              value )
```

```
The setValue function sets the value of the object.


Args:
self: Represent the instance of the class
value: Set the value of the instance variable __value

Returns:
The value that was passed to it

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 22 of file DataTransfer.py.

```
    def setValue(self, value):
        """
    The setValue function sets the value of the object.


    Args:
        self: Represent the instance of the class
        value: Set the value of the instance variable __value

    Returns:
        The value that was passed to it

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__value = value

```

3.8.3.3 whileValue()

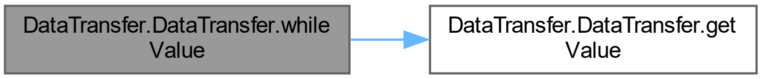
```
def DataTransfer.DataTransfer.whileValue (
                self )
```

```
The whileValue function is a function that will run the getValue function until it is told to stop.
This allows for the program to constantly be checking for new values from the sensor.

Args:
self: Refer to the current instance of the class

Returns:
The value of the input

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 55 of file DataTransfer.py.

```
    def whileValue(self):
        """
    The whileValue function is a function that will run the getValue function until it is told to stop.
    This allows for the program to constantly be checking for new values from the sensor.

    Args:
        self: Refer to the current instance of the class

    Returns:
        The value of the input

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            self.getValue()
```

Here is the call graph for this function:


DataTransfer.DataTransfer.while Value
DataTransfer.DataTransfer.get Value

3.8.4 Member Data Documentation

3.8.4.1 __value

`DataTransfer.DataTransfer.__value [private]`

Definition at line 20 of file DataTransfer.py.

The documentation for this class was generated from the following file:

- DataTransfer.py

3.9 FileSaver.FileSaver Class Reference

Public Member Functions

- def __init__ (self, method, outputDF, AppendingPath=None)
- def getPath (self)

Public Attributes

- docPath
- data
- dataAppending
- appendFlag
- fileName
- uiFlag
- primaryKey
- outputFrame

3.9.1 Detailed Description

Definition at line 13 of file FileSaver.py.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 __init__()

```
def FileSaver.FileSaver.__init__ (
              self,
              method,
              outputDF,
              AppendingPath = None )
```

```
The __init__ function is called when the class is instantiated.
It sets up the instance of the class, and defines all variables that will be used by other functions in this clas
The __init__ function takes two arguments: self and method.  The first argument, self, refers to an instance of a
class (in this case it's an instance of DataFrameSaver). The second argument, method refers to a string value tha
is passed into DataFrameSaver when it's instantiated.

Args:
self: Represent the instance of the class
method: Determine which dataframe to append the new data to
outputDF: Pass in the dataframe that will be saved to a csv file
AppendingPath: Specify the path to an existing csv file that you want to append your dataframe to

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 15 of file FileSaver.py.

```
00015     def __init__(self, method, outputDF, AppendingPath=None):
00016         """
00017     The __init__ function is called when the class is instantiated.
00018     It sets up the instance of the class, and defines all variables that will be used by other functions
      in this class.
00019     The __init__ function takes two arguments: self and method.  The first argument, self, refers to an
      instance of a
00020     class (in this case it's an instance of DataFrameSaver). The second argument, method refers to a
      string value that
00021     is passed into DataFrameSaver when it's instantiated.
00022 
00023     Args:
00024         self: Represent the instance of the class
00025         method: Determine which dataframe to append the new data to
00026         outputDF: Pass in the dataframe that will be saved to a csv file
00027         AppendingPath: Specify the path to an existing csv file that you want to append your dataframe to
00028 
00029     Returns:
00030         Nothing
00031 
00032     Doc Author:
00033         Willem van der Schans, Trelent AI
00034     """
00035         self.docPath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
00036             datetime.datetime.today().strftime('%m%d%Y'))
00037         self.data = outputDF
```

```
        self.dataAppending = None
        self.appendFlag = True
        self.fileName = f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
        self.uiFlag = True

        if method.lower() == "ure":
            self.primaryKey = "ListingKeyNumeric"
        elif method.lower() == "cm":
            self.primaryKey = "id"
        elif "realtor" in method.lower():
            self.primaryKey = None
            self.uiFlag = False
        elif method.lower() == "cfbp":
            self.primaryKey = None
            self.uiFlag = False
        else:
            raise ValueError("method input is invalid choice one of 4 options: URE, CM, Realtor, CFBP")

        if AppendingPath is None:
            self.appendFlag = False
        else:
            self.dataAppending = pd.read_csv(AppendingPath)

        if self.appendFlag:
            if self.primaryKey is not None:
                # Due to low_memory loading the columns are not typed properly,
                # since we are comparing this will be an issue since we need to do type comparisons,
                # so here we coerce the types of the primary keys to numeric.
                # If another primary key is ever chosen make sure to core to the right data type.
                self.dataAppending[self.primaryKey] = pd.to_numeric(self.dataAppending[self.primaryKey])
                self.data[self.primaryKey] = pd.to_numeric(self.data[self.primaryKey])

                self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(subset=[self.primaryKey],
                                                                                              keep="last")
            else:
                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(keep="last")
        else:
            self.outputFrame = self.data

        if os.path.exists(self.docPath):
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
        else:
            os.mkdir(self.docPath)
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)

        if self.uiFlag:
            if self.appendFlag:
                PopupWrapped(text=f"File Appended and Saved to {self.docPath.joinpath(self.fileName)}",
                             windowType="savedLarge")

                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | {method} API
request Completed | File Appended and Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0")
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Appending
Statistics | Method: {method} | Appending file rows: {self.dataAppending.shape[0]}, Total Rows:
{(self.dataAppending.shape[0] + self.data.shape[0])}, Duplicates Dropped {(self.dataAppending.shape[0] +
self.data.shape[0])-self.outputFrame.shape[0]}")
            else:
                PopupWrapped(text=f"File Saved to {self.docPath.joinpath(self.fileName)}",
windowType="savedLarge")

                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | {method} API
request Completed | File Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0")
        else:
            pass

```

3.9.3 Member Function Documentation

3.9.3.1 getPath()

```
def FileSaver.FileSaver.getPath (
              self )
```

```
The getPath function returns the path to the file.
It is a string, and it joins the docPath with the fileName.

Args:
self: Represent the instance of the class

Returns:
The path to the file

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 101 of file FileSaver.py.

```
    def getPath(self):
        """
    The getPath function returns the path to the file.
        It is a string, and it joins the docPath with the fileName.

    Args:
        self: Represent the instance of the class

    Returns:
        The path to the file

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        return str(self.docPath.joinpath(self.fileName))
```

3.9.4 Member Data Documentation

3.9.4.1 appendFlag

```
FileSaver.FileSaver.appendFlag
```

Definition at line 39 of file FileSaver.py.

3.9.4.2 data

```
FileSaver.FileSaver.data
```

Definition at line 37 of file FileSaver.py.

3.9.4.3 dataAppending

`FileSaver.FileSaver.dataAppending`

Definition at line 38 of file FileSaver.py.

3.9.4.4 docPath

`FileSaver.FileSaver.docPath`

Definition at line 35 of file FileSaver.py.

3.9.4.5 fileName

`FileSaver.FileSaver.fileName`

Definition at line 40 of file FileSaver.py.

3.9.4.6 outputFrame

`FileSaver.FileSaver.outputFrame`

Definition at line 70 of file FileSaver.py.

3.9.4.7 primaryKey

`FileSaver.FileSaver.primaryKey`

Definition at line 44 of file FileSaver.py.

3.9.4.8 uiFlag

`FileSaver.FileSaver.uiFlag`

Definition at line 41 of file FileSaver.py.

The documentation for this class was generated from the following file:

- FileSaver.py

3.10 API_Calls.Initializer.initializer Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- classObj

Private Member Functions

- def __ShowGui (self, layout, text)
- def __CreateFrame (self)

3.10.1 Detailed Description

Definition at line 22 of file Initializer.py.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 __init__()

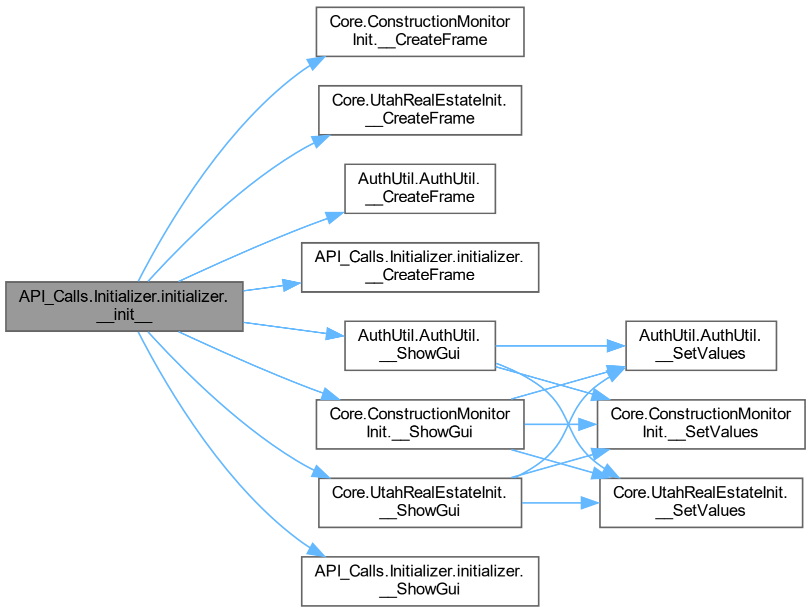
```
def API_Calls.Initializer.initializer.__init__ (
              self )
```

```
The __init__ function is called when the class is instantiated.
It sets up the logging, calls the __ShowGui function to create and display
the GUI, and then calls __CreateFrame to create a frame for displaying widgets.


Args:
self: Represent the instance of the class

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 24 of file Initializer.py.

```
00024     def __init__(self):
00025 
00026         """
00027     The __init__ function is called when the class is instantiated.
00028     It sets up the logging, calls the __ShowGui function to create and display
00029     the GUI, and then calls __CreateFrame to create a frame for displaying widgets.
00030 
00031 
00032     Args:
00033         self: Represent the instance of the class
00034 
00035     Returns:
00036         Nothing
00037 
00038     Doc Author:
00039         Willem van der Schans, Trelent AI
00040     """
00041         self.classObj = None
00042 
00043         logger()
00044 
00045         print("\n\n-------------Initiate Program---------------------\n\n")
00046 
00047         self.__ShowGui(self.__CreateFrame(), "Data Tool")
00048 
00049         print("\n\n-------------Closing Program---------------------\n\n")
00050 
```

Here is the call graph for this function:


Core.ConstructionMonitor Init.__CreateFrame
Core.UtahRealEstateInit. __CreateFrame
AuthUtil.AuthUtil. __CreateFrame
API_Calls.Initializer.initializer. __CreateFrame
API_Calls.Initializer.initializer. __init__
AuthUtil.AuthUtil. __ShowGui
AuthUtil.AuthUtil. __SetValues
Core.ConstructionMonitor Init.__ShowGui
Core.ConstructionMonitor Init.__SetValues
Core.UtahRealEstateInit. __ShowGui
Core.UtahRealEstateInit. __SetValues
API_Calls.Initializer.initializer. __ShowGui

3.10.3 Member Function Documentation

3.10.3.1 __CreateFrame()

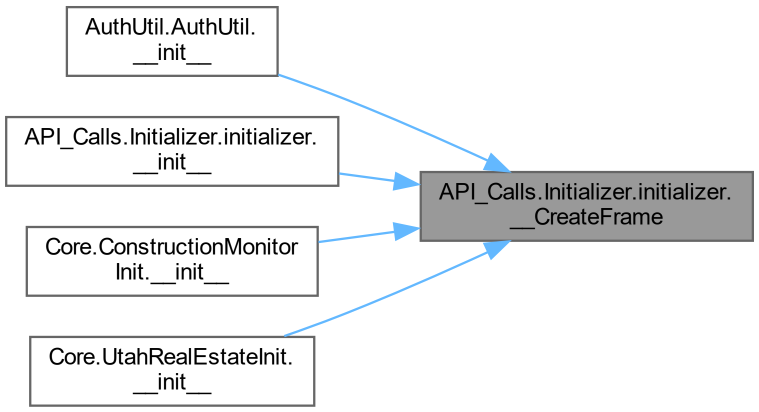
```
def API_Calls.Initializer.initializer.__CreateFrame (
              self )  [private]
```

```
The __CreateFrame function is a helper function that creates the layout for the main window.
It returns a list of lists, which is then passed to sg.Window() as its layout parameter.

Args:
self: Represent the instance of the class

Returns:
A list of lists, which is then passed to the sg

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 136 of file Initializer.py.

```
    def __CreateFrame(self):

        """
    The __CreateFrame function is a helper function that creates the layout for the main window.
    It returns a list of lists, which is then passed to sg.Window() as its layout parameter.

    Args:
        self: Represent the instance of the class

    Returns:
        A list of lists, which is then passed to the sg

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line0 = [sg.HSeparator()]

        line1 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line3 = [sg.HSeparator()]

        line4 = [sg.Push(),
                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)), sg.Push(),
                  sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]

        line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(),
                  sg.Button("CFPB Mortgage", size=(20, None)),
                  sg.Push()]]

        line8 = [sg.HSeparator()]

        line9 = [sg.Push(),
                 sg.Text("Utilities", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
                   sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]

        line11 = [sg.HSeparator()]

        layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10, line11]

        return layout
```

Here is the caller graph for this function:



3.10.3.2 __ShowGui()

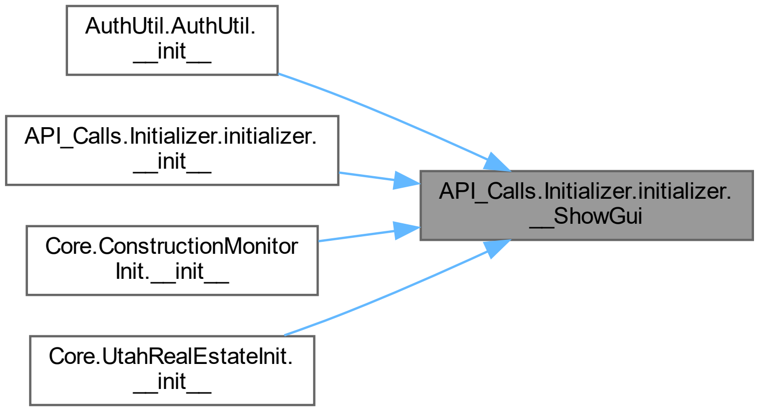
```
def API_Calls.Initializer.initializer.__ShowGui (
              self,
              layout,
              text )  [private]
```

```
The __ShowGui function is the main function that displays the GUI.
It takes two arguments: layout and text. Layout is a list of lists, each containing a tuple with three elements:
1) The type of element to be displayed (e.g., &quot;Text&quot;, &quot;InputText&quot;, etc.)
2) A dictionary containing any additional parameters for that element (e.g., size, default value, etc.)
3) An optional key name for the element (used in event handling). If no key name is provided then one will be gen

Args:
self: Represent the instance of the class
layout: Pass the layout of the window to be created
text: Set the title of the window

Returns:
A window object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 51 of file Initializer.py.

```
00051     def __ShowGui(self, layout, text):
00052 
00053         """
00054     The __ShowGui function is the main function that displays the GUI.
00055     It takes two arguments: layout and text. Layout is a list of lists, each containing a tuple with three
      elements:
        1) The type of element to be displayed (e.g., &quot;Text&quot;, &quot;InputText&quot;, etc.)
        2) A dictionary containing any additional parameters for that element (e.g., size, default value, etc.)
        3) An optional key name for the element (used in event handling). If no key name is provided then one will be generated automatically by PySimpleGUIQt based on its position in the layout list

    Args:
        self: Represent the instance of the class
        layout: Pass the layout of the window to be created
        text: Set the title of the window

    Returns:
        A window object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        # Todo Gitlab Update
        versionChecker()

        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Construction Monitor":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Construction Monitor API Call-----------------")
                ConstructionMonitorMain(ConstructionMonitorInit())
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Construction Monitor API Call---------------------\n")
            elif event == "Utah Real Estate":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Utah Real Estate API Call-----------------")
                UtahRealEstateMain(UtahRealEstateInit())
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Utah Real Estate API Call---------------------\n")
            elif event == "Realtor.Com":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Realtor.com API Call-----------------")
                realtorCom()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Realtor.com API Call---------------------\n")
            elif event == "CFPB Mortgage":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating ffiec.cfpb API Call-----------------")
                CFBP()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing ffiec.cfpb API Call---------------------\n")
            elif event == "Authorization Utility":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Initiating Authorization Utility-----------------")
                AuthUtil()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Closing Authorization Utility---------------------\n")
            elif event == "Open Data Folder":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | -------------Data Folder Opened-----------------")
                try:
                    os.system(f"start {Path(os.path.expanduser('~/Documents')).joinpath('GardnerUtilData')}")
                except:
                    try:
                        os.system(f"start {Path(os.path.expanduser('~/Documents'))}")
                    except Exception as e:
                        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Initializer.py | Error = {e} | Documents folder not found")
                        PopupWrapped(
                            text="Documents folder not found. Please create a Windows recognized documents folder",
                            windowType="errorLarge")

            elif event in ('Exit', None):
                try:
                    break
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Initializer.py | Error = {e} | Error on program exit, for logging purposes only.")
                    break
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

```

Here is the caller graph for this function:



3.10.4 Member Data Documentation

3.10.4.1 classObj

`API_Calls.Initializer.initializer.classObj`

Definition at line 41 of file Initializer.py.

The documentation for this class was generated from the following file:

- Initializer.py

3.11 PopupWrapped.PopupWrapped Class Reference

Public Member Functions

- def __init__ (self, text="", windowType="notice", error=None)
- def stopWindow (self)
- def textUpdate (self, sleep=0.5)
- def windowPush (self)
- def openFile (self)

Private Member Functions

- def __createLayout (self)
- def __createWindow (self)

Private Attributes

- __text
- __type
- __error
- __layout
- __windowObj
- __thread
- __counter
- __docpath
- __errorFlag

3.11.1 Detailed Description

Definition at line 15 of file PopupWrapped.py.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 __init__()

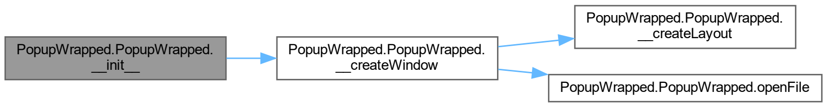
```
def PopupWrapped.PopupWrapped.__init__ (
                self,
                text = "",
                windowType = "notice",
                error = None )
```

```
The __init__ function is the first function that gets called when an object of this class is created.
It sets up all the variables and creates a window for us to use.
Args:
self: Represent the instance of the class
text: Set the text of the window
windowType: Determine what type of window to create
error: Display the error message in the window
Returns:
Nothing
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 17 of file PopupWrapped.py.

```
00017     def __init__(self, text="", windowType="notice", error=None):
00018         """
00019     The __init__ function is the first function that gets called when an object of this class is created.
00020     It sets up all the variables and creates a window for us to use.
00021     Args:
00022         self: Represent the instance of the class
00023         text: Set the text of the window
00024         windowType: Determine what type of window to create
00025         error: Display the error message in the window
00026     Returns:
00027         Nothing
00028     Doc Author:
00029         Willem van der Schans, Trelent AI
00030     """
00031         self.__text = text
00032         self.__type = windowType
00033         self.__error = error
00034         self.__layout = []
00035         self.__windowObj = None
00036         self.__thread = None
00037         self.__counter = 0
00038         self.__docpath = None
00039         self.__errorFlag = False
00040 
00041         try:
00042             if "File Appended and Saved to " in self.__text:
00043                 self.__docpath = str(self.__text[27:])
00044             elif "File Saved to " in self.__text:
00045                 self.__docpath = str(self.__text[14:])
00046             else:
00047                 pass
00048         except Exception as e:
00049             if self.__type == "savedLarge":
00050                 print(
00051                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | PopupWrapped.py
      | Error = {e} | Error creating self.__docpath open file button not available")
00052                 self.__errorFlag = True
00053             else:
00054                 pass
00055 
00056         self.__createWindow()
00057 
```

Here is the call graph for this function:


PopupWrapped.PopupWrapped.__init__
PopupWrapped.PopupWrapped.__createWindow
PopupWrapped.PopupWrapped.__createLayout
PopupWrapped.PopupWrapped.openFile

3.11.3 Member Function Documentation

3.11.3.1 __createLayout()

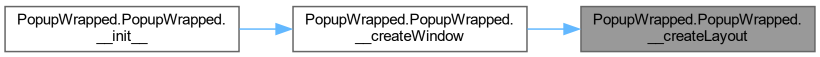
```
def PopupWrapped.PopupWrapped.__createLayout (
              self )  [private]
```

```
The __createLayout function is used to create the layout of the window.
The function takes class variables and returns a window layout.
It uses a series of if statements to determine what type of window it is, then creates a layout based on that info
Args:
self: Refer to the current instance of a class
Returns:
A list of lists
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 58 of file PopupWrapped.py.

```
    def __createLayout(self):
        """
    The __createLayout function is used to create the layout of the window.
    The function takes class variables and returns a window layout.
    It uses a series of if statements to determine what type of window it is, then creates a layout based
      on that information.
    Args:
        self: Refer to the current instance of a class
    Returns:
        A list of lists
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')
        __Line1 = None
        __Line2 = None

        if self.__type == "notice":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "noticeLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "savedLarge":
            if self.__errorFlag:
                __Line1 = [sg.Push(),
                           sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                           sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
                __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
            else:
                __Line1 = [sg.Push(),
                           sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                           sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
                __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "errorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "FatalErrorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="left", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "error":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}: {self.__error}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "AuthError":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool [Web Browser]"),
                       sg.Ok(button_text="Return", focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "versionWindow":
            __Line1 = [sg.Push(),
                       sg.Text(f"{self.__text}", justification="left", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Button(button_text="Download"),
                       sg.Ok(button_text="Continue", focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "progress":
            __Line1 = [sg.Push(),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]

        if self.__type == "progress":
            self.__layout = [__Line1, ]
        else:
            self.__layout = [__Line1, __Line2]

```

Here is the caller graph for this function:


PopupWrapped.PopupWrapped.__init__
PopupWrapped.PopupWrapped.__createWindow
PopupWrapped.PopupWrapped.__createLayout

3.11.3.2 __createWindow()

```
def PopupWrapped.PopupWrapped.__createWindow (
                self ) [private]
```

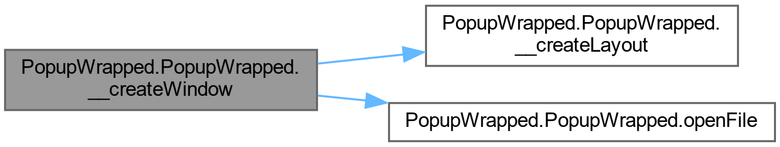
```
The __createWindow function is used to create the window object that will be displayed.
The function takes class variables and a window object. The function first calls __createLayout, which creates th
```

```
Args:
self: Reference the instance of the class
Returns:
A window object
Doc Author:
Willem van der Schans, Trelent AI
```

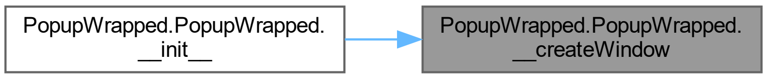
Definition at line 133 of file PopupWrapped.py.

```
00133     def __createWindow(self):
00134         """
00135     The __createWindow function is used to create the window object that will be displayed.
00136     The function takes class variables and a window object. The function first calls __createLayout, which
      creates the layout for the window based on what type of message it is (error, notice, progress). Then it
      uses PySimpleGUI's Window class to create a new window with that layout and some other parameters such as
      title and icon. If this is not a progress bar or permanent message then we start a timer loop that waits
      until either 100 iterations have passed or an event has been triggered (such as clicking &quot;Ok&quot; or
      closing the window). Once one of these events occurs
00137     Args:
00138         self: Reference the instance of the class
00139     Returns:
00140         A window object
00141     Doc Author:
00142         Willem van der Schans, Trelent AI
00143     """
00144         self.__createLayout()
00145 
00146         if self.__type == "progress":
00147             self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout,
      finalize=True,
00148                                          modal=True,
00149                                          keep_on_top=True,
00150                                          disable_close=False,
00151                                          icon=ImageLoader("taskbar_icon.ico"),
00152                                          size=(290, 50))
00153         elif self.__type == "noticeLarge":
00154             self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
00155                                          modal=True,
00156                                          keep_on_top=True,
00157                                          disable_close=False,
00158                                          icon=ImageLoader("taskbar_icon.ico"))
00159         elif self.__type == "savedLarge":
00160             self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
00161                                          modal=True,
00162                                          keep_on_top=False,
00163                                          disable_close=False,
00164                                          icon=ImageLoader("taskbar_icon.ico"))
00165         elif self.__type == "errorLarge":
00166             self.__windowObj = sg.Window(title="Error", layout=self.__layout, finalize=True,
00167                                          modal=True,
00168                                          keep_on_top=True,
00169                                          disable_close=False,
00170                                          icon=ImageLoader("taskbar_icon.ico"))
00171         elif self.__type == "FatalErrorLarge":
00172             self.__windowObj = sg.Window(title="Fatal Error", layout=self.__layout, finalize=True,
00173                                          modal=True,
00174                                          keep_on_top=True,
00175                                          disable_close=False,
00176                                          icon=ImageLoader("taskbar_icon.ico"))
00177         elif self.__type == "AuthError":
00178             self.__windowObj = sg.Window(title="Authentication Error", layout=self.__layout,
      finalize=True,
00179                                          modal=True,
00180                                          keep_on_top=True,
00181                                          disable_close=False,
00182                                          icon=ImageLoader("taskbar_icon.ico"))
00183         elif self.__type == "versionWindow":
00184             self.__windowObj = sg.Window(title="Update Notice", layout=self.__layout, finalize=True,
00185                                          modal=True,
00186                                          keep_on_top=True,
00187                                          disable_close=False,
00188                                          icon=ImageLoader("taskbar_icon.ico"))
00189         else:
00190             self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout,
      finalize=True,
00191                                          modal=True,
00192                                          keep_on_top=True,
00193                                          disable_close=False,
00194                                          icon=ImageLoader("taskbar_icon.ico"),
                                          size=(290, 80))

        if self.__type != "progress" or self.__type.startswith("perm"):
            timer = 0
            while timer < 100:
                event, values = self.__windowObj.read()
                if event == "Ok" or event == sg.WIN_CLOSED or event == "Return" or event == "Continue":
                    break
                elif event == "Open Generation Tool [Web Browser]":
                    webbrowser.open(settings.settingGenerationToolLink, new=2, autoraise=True)
                    pass
                elif event == "Open File":
                    threadFile = threading.Thread(target=self.openFile,
                                                  daemon=False)
                    threadFile.start()
                    time.sleep(3)
                    break
                elif event == "Download":
                    # Todo Gitlab Update
                    webbrowser.open(settings.settingDownloadSourceLink, new=2,
                                    autoraise=True)
                    pass
                time.sleep(0.1)

            if self.__type == "FatalErrorLarge":
                try:
                    os.system(
                        f"start
{Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}")
                except Exception as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
PopupWrapped.py | Error = {e} | Log Folder not found please search manually for
%APPDATA%\Roaming\GardnerUtil\Logs\n")

            self.__windowObj.close()

```

Here is the call graph for this function:


PopupWrapped.PopupWrapped.__createWindow
PopupWrapped.PopupWrapped.__createLayout
PopupWrapped.PopupWrapped.openFile

Here is the caller graph for this function:


PopupWrapped.PopupWrapped.__init__
PopupWrapped.PopupWrapped.__createWindow

3.11.3.3 openFile()

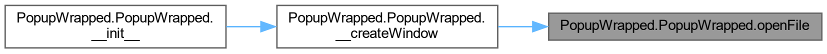
```
def PopupWrapped.PopupWrapped.openFile (
              self )
```

```
The openFile function opens the file that is associated with the
document object.  It does this by calling os.system and passing it
self.__docpath as an argument.

Args:
self: Represent the instance of the object itself

Returns:
The filepath of the document

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 290 of file PopupWrapped.py.

```
    def openFile(self):
        """
    The openFile function opens the file that is associated with the
        document object.  It does this by calling os.system and passing it
        self.__docpath as an argument.

    Args:
        self: Represent the instance of the object itself

    Returns:
        The filepath of the document

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        os.system(self.__docpath)
```

Here is the caller graph for this function:


PopupWrapped.PopupWrapped.__init__
PopupWrapped.PopupWrapped.__createWindow
PopupWrapped.PopupWrapped.openFile

3.11.3.4 stopWindow()

```
def PopupWrapped.PopupWrapped.stopWindow (
              self )
```

```
The stopWindow function is used to close the window object that was created in the startWindow function.
This is done by calling the close() method on self.__windowObj, which will cause it to be destroyed.
Args:
self: Represent the instance of the class
Returns:
The window object
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 229 of file PopupWrapped.py.

```
    def stopWindow(self):
        """
    The stopWindow function is used to close the window object that was created in the startWindow
 function.
    This is done by calling the close() method on self.__windowObj, which will cause it to be destroyed.
    Args:
        self: Represent the instance of the class
    Returns:
        The window object
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__windowObj.close()

```

3.11.3.5 textUpdate()

```
def PopupWrapped.PopupWrapped.textUpdate (
                self,
                sleep = 0.5 )
```

```
The textUpdate function is a function that updates the text in the text field.
It does this by adding dots to the end of it, and then removing them. This creates
a loading effect for when something is being processed.
Args:
self: Refer to the object itself
sleep: Control the speed of the text update
Returns:
A string that is the current text of the text field
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 242 of file PopupWrapped.py.

```
    def textUpdate(self, sleep=0.5):
        """
    The textUpdate function is a function that updates the text in the text field.
    It does this by adding dots to the end of it, and then removing them. This creates
    a loading effect for when something is being processed.
    Args:
        self: Refer to the object itself
        sleep: Control the speed of the text update
    Returns:
        A string that is the current text of the text field
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__counter += 1
        if self.__counter == 4:
            self.__counter = 1
        newString = ""
        if self.__type == "notice":
            pass
        elif self.__type == "error":
            pass
        elif self.__type == "progress":
            newString = f"{self.__text}{'.' * self.__counter}"
        self.__windowObj.write_event_value('update-textField-', newString)

        time.sleep(sleep)

```

3.11.3.6 windowPush()

```
def PopupWrapped.PopupWrapped.windowPush (
              self )
```

```
The windowPush function is used to update the values of a window object.
The function takes in an event and values from the window object, then checks if the event starts with 'update'.
If it does, it will take everything after 'update' as a key for updating that specific value.
It will then update that value using its key and refresh the window.
Args:
self: Reference the object that is calling the function
Returns:
A tuple containing the event and values
Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 269 of file PopupWrapped.py.

```
    def windowPush(self):

        """
    The windowPush function is used to update the values of a window object.
        The function takes in an event and values from the window object, then checks if the event starts
 with 'update'.
        If it does, it will take everything after 'update' as a key for updating that specific value.
        It will then update that value using its key and refresh the window.
    Args:
        self: Reference the object that is calling the function
    Returns:
        A tuple containing the event and values
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        event, values = self.__windowObj.read()

        if event.startswith('update'):
            __key_to_update = event[len('update'):]
            self.__windowObj[__key_to_update].update(values[event])
            self.__windowObj.refresh()

```

3.11.4 Member Data Documentation

3.11.4.1 __counter

```
PopupWrapped.PopupWrapped.__counter  [private]
```

Definition at line 37 of file PopupWrapped.py.

3.11.4.2 __docpath

```
PopupWrapped.PopupWrapped.__docpath  [private]
```

Definition at line 38 of file PopupWrapped.py.

3.11.4.3 __error

`PopupWrapped.PopupWrapped.__error [private]`

Definition at line 33 of file PopupWrapped.py.

3.11.4.4 __errorFlag

`PopupWrapped.PopupWrapped.__errorFlag [private]`

Definition at line 39 of file PopupWrapped.py.

3.11.4.5 __layout

`PopupWrapped.PopupWrapped.__layout [private]`

Definition at line 34 of file PopupWrapped.py.

3.11.4.6 __text

`PopupWrapped.PopupWrapped.__text [private]`

Definition at line 31 of file PopupWrapped.py.

3.11.4.7 __thread

`PopupWrapped.PopupWrapped.__thread [private]`

Definition at line 36 of file PopupWrapped.py.

3.11.4.8 __type

`PopupWrapped.PopupWrapped.__type [private]`

Definition at line 32 of file PopupWrapped.py.

3.11.4.9 __windowObj

`PopupWrapped.PopupWrapped.__windowObj [private]`

Definition at line 35 of file PopupWrapped.py.

The documentation for this class was generated from the following file:

- PopupWrapped.py

3.12 Core.realtorCom Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- dfState
- dfCounty
- dfZip
- uiString

Private Member Functions

- def __showUi (self)
- def __linkGetter (self)
- def __dataUpdater (self)

Private Attributes

- __page_html
- __update_date
- __last_date
- __idDict
- __linkDict

3.12.1 Detailed Description

Definition at line 16 of file Realtor/Core.py.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 __init__()

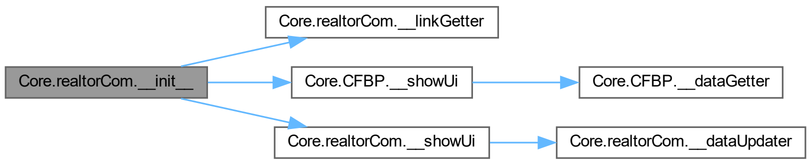
```
def Core.realtorCom.__init__ (
              self )
```

```
The __init__ function is called when the class is instantiated.
It sets up the initial state of an object, and it’s where you put code that needs to run before anything else in

Args:
self: Represent the instance of the class

Returns:
A new object

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 18 of file Realtor/Core.py.

```
00018     def __init__(self):
00019         """
00020     The __init__ function is called when the class is instantiated.
00021     It sets up the initial state of an object, and it’s where you put code that needs to run before
      anything else in your class.
00022
00023     Args:
00024         self: Represent the instance of the class
00025
00026     Returns:
00027         A new object
00028
00029     Doc Author:
00030         Willem van der Schans, Trelent AI
00031     """
00032         self.__page_html = None
00033         self.__update_date = None
00034         self.__last_date = None
00035         self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00036         self.__linkDict = {}
00037         self.dfState = None
00038         self.dfCounty = None
00039         self.dfZip = None
00040         self.uiString = "Files Saved to \n"
00041
00042         eventReturn = confirmDialog()
00043         if eventReturn == "Continue":
00044             page_html = requests.get(settings.settingRealtorLink).text
00045             self.__page_html = BeautifulSoup(page_html, "html.parser")
00046             startTime = datetime.datetime.now().replace(microsecond=0)
00047             self.__linkGetter()
00048             print(
00049                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Link Dictionary =
      {self.__idDict}")
00050             self.__showUi()
00051             PopupWrapped(text=self.uiString, windowType="noticeLarge")
00052             print(
00053                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Data retrieved with
      in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) -
      startTime).total_seconds()))}")
00054         else:
00055             print(
00056                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled
      Request")
00057             pass
00058
```

Here is the call graph for this function:



3.12.3 Member Function Documentation

3.12.3.1 __dataUpdater()

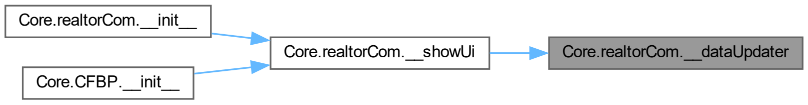
```
def Core.realtorCom.__dataUpdater (
              self )  [private]
```

```
The __dataUpdater function is a private function that updates the dataframes for each of the three
types of realtor data. It takes class variables and return the path to the saved file. The function first creates
dictionary called tempdf, then iterates through each key in self.__idDict (which contains all three ids).
For each key, it reads in a csv file from the link associated with that id and saves it to tempdf as a pandas
DataFrame object. Then, depending on which type of realtor data we are dealing with (State/County/Zip), we save


Args:
self: Access the attributes and methods of the class

Returns:
The path of the saved file

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 114 of file Realtor/Core.py.

```
    def __dataUpdater(self):

        """
    The __dataUpdater function is a private function that updates the dataframes for each of the three
        types of realtor data. It takes class variables and return the path to the saved file. The
 function first creates an empty
        dictionary called tempdf, then iterates through each key in self.__idDict (which contains all
 three ids).
        For each key, it reads in a csv file from the link associated with that id and saves it to tempdf
 as a pandas
        DataFrame object. Then, depending on which type of realtor data we are dealing with
 (State/County/Zip), we save


    Args:
        self: Access the attributes and methods of the class

    Returns:
        The path of the saved file

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        for key, value in self.__idDict.items():
            tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)

            if key == "State":
                self.dfState = tempdf
            elif key == "County":
                self.dfCounty = tempdf
            elif key == "Zip":
                self.dfZip = tempdf

            FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
            self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()} \n"
```

Here is the caller graph for this function:



3.12.3.2 __linkGetter()

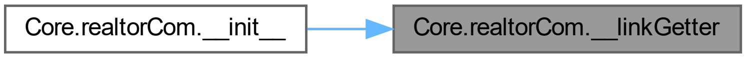
```
def Core.realtorCom.__linkGetter (
              self ) [private]
```

```
The __linkGetter function is a private function that takes the idDict dictionary and adds
a link to each entry in the dictionary. The link is used to access historical data for each
scope symbol.

Args:
self: Refer to the object itself

Returns:
A dictionary of all the links to the history pages

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 87 of file Realtor/Core.py.

```
    def __linkGetter(self):

        """
    The __linkGetter function is a private function that takes the idDict dictionary and adds
    a link to each entry in the dictionary. The link is used to access historical data for each
    scope symbol.

    Args:
        self: Refer to the object itself

    Returns:
        A dictionary of all the links to the history pages

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        for key, value in self.__idDict.items():
            for row in self.__page_html.find_all("div", {"class": "monthly"}):
                try:
                    for nestedRow in row.find_all("a"):
                        if "History" in str(nestedRow.get("href")) and key in str(nestedRow.get("href")):
                            self.__idDict[key] = {"id": value, "link": nestedRow.get("href")}
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
      Realtor/Core.py | Error = {e} | Error while getting document links for realtor.com")
                    RESTError(801)
                    raise SystemExit(801)

```

Here is the caller graph for this function:



3.12.3.3 __showUi()

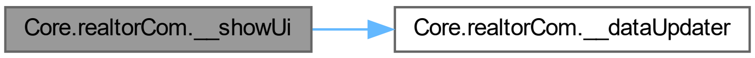
```
def Core.realtorCom.__showUi (
              self )  [private]
```

```
The __showUi function is a helper function that creates and displays the progress window.
It also starts the dataUpdater thread, which will update the progress bar as it runs.


Args:
self: Represent the instance of the class

Returns:
A popupwrapped object

Doc Author:
Willem van der Schans, Trelent AI
```

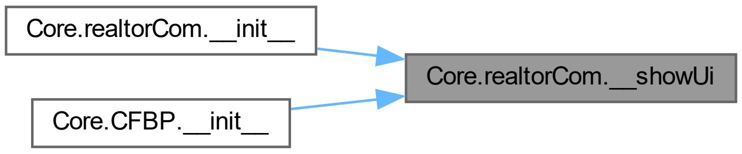
Definition at line 59 of file Realtor/Core.py.

```
    def __showUi(self):

        """
    The __showUi function is a helper function that creates and displays the progress window.
    It also starts the dataUpdater thread, which will update the progress bar as it runs.


    Args:
        self: Represent the instance of the class

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__dataUpdater,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()

```

Here is the call graph for this function:



Here is the caller graph for this function:



3.12.4 Member Data Documentation

3.12.4.1 __idDict

`Core.realtorCom.__idDict [private]`

Definition at line 35 of file Realtor/Core.py.

3.12.4.2 __last_date

`Core.realtorCom.__last_date [private]`

Definition at line 34 of file Realtor/Core.py.

3.12.4.3 __linkDict

`Core.realtorCom.__linkDict [private]`

Definition at line 36 of file Realtor/Core.py.

3.12.4.4 __page_html

`Core.realtorCom.__page_html [private]`

Definition at line 32 of file Realtor/Core.py.

3.12.4.5 __update_date

`Core.realtorCom.__update_date [private]`

Definition at line 33 of file Realtor/Core.py.

3.12.4.6 dfCounty

`Core.realtorCom.dfCounty`

Definition at line 38 of file Realtor/Core.py.

3.12.4.7 dfState

`Core.realtorCom.dfState`

Definition at line 37 of file Realtor/Core.py.

3.12.4.8 dfZip

```
Core.realtorCom.dfZip
```

Definition at line 39 of file Realtor/Core.py.

3.12.4.9 uiString

```
Core.realtorCom.uiString
```

Definition at line 40 of file Realtor/Core.py.

The documentation for this class was generated from the following file:

- Realtor/Core.py

3.13 Settings.settings Class Reference

Static Public Attributes

- str settingVersion = "1.2.0"
- str settingGithubApiUrl = "https://api.github.com/repos/Kydoimos97/GardnerApiUtility/releases/latest"
- str settingGenerationToolLink = 'https://www.debugbear.com/basic-auth-header-generator'
- str settingDownloadSourceLink = 'https://github.com/Kydoimos97/GardnerApiUtility/releases/latest'
- str settingCFBPLink = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"
- str settingCMRestDomain = "https://api.constructionmonitor.com/v2/powersearch/?"
- str settingRealtorLink = "https://www.realtor.com/research/data/"
- str settingURERestDomain = "https://resoapi.utahrealestate.com/reso/odata/Property?"

3.13.1 Detailed Description

Definition at line 6 of file Settings.py.

3.13.2 Member Data Documentation

3.13.2.1 settingCFBPLink

```
str Settings.settings.settingCFBPLink = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"
[static]
```

Definition at line 21 of file Settings.py.

3.13.2.2 settingCMRestDomain

```
str Settings.settings.settingCMRestDomain = "https://api.constructionmonitor.com/v2/powersearch/?"
[static]
```

Definition at line 25 of file Settings.py.

3.13.2.3 settingDownloadSourceLink

```
str Settings.settings.settingDownloadSourceLink = 'https://github.com/Kydoimos97/GardnerApi↩
Utility/releases/latest'  [static]
```

Definition at line 17 of file Settings.py.

3.13.2.4 settingGenerationToolLink

```
str Settings.settings.settingGenerationToolLink = 'https://www.debugbear.com/basic-auth-header-generator'
[static]
```

Definition at line 15 of file Settings.py.

3.13.2.5 settingGithubApiUrl

```
str Settings.settings.settingGithubApiUrl = "https://api.github.com/repos/Kydoimos97/GardnerApi↩
Utility/releases/latest"  [static]
```

Definition at line 11 of file Settings.py.

3.13.2.6 settingRealtorLink

```
str Settings.settings.settingRealtorLink = "https://www.realtor.com/research/data/"  [static]
```

Definition at line 29 of file Settings.py.

3.13.2.7 settingURERestDomain

```
str Settings.settings.settingURERestDomain = "https://resoapi.utahrealestate.com/reso/odata/Property?"
[static]
```

Definition at line 33 of file Settings.py.

3.13.2.8 settingVersion

```
str Settings.settings.settingVersion = "1.2.0"  [static]
```

Definition at line 9 of file Settings.py.

The documentation for this class was generated from the following file:

- Settings.py

3.14 Core.UtahRealEstateInit Class Reference

Public Member Functions

- def __init__ (self)

Public Attributes

- StandardStatus
- ListedOrModified
- dateStart
- dateEnd
- select
- file_name
- append_file

Private Member Functions

- def __ShowGui (self, layout, text)
- def __SetValues (self, values)

Static Private Member Functions

- def __CreateFrame ()

3.14.1 Detailed Description

Definition at line 25 of file UtahRealEstate/Core.py.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 __init__()

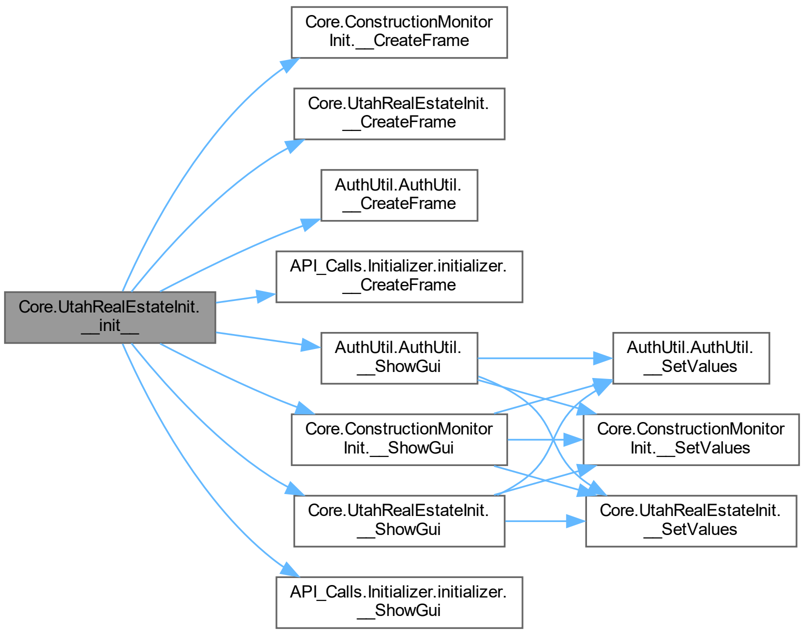
```
def Core.UtahRealEstateInit.__init__ (
              self )
```

```
The __init__ function is called when the class is instantiated.
It sets up the initial state of the object.


Args:
self: Represent the instance of the class

Returns:
The __createframe function

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 27 of file UtahRealEstate/Core.py.

```
    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the initial state of the object.


    Args:
        self: Represent the instance of the class

    Returns:
        The __createframe function

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.dateStart = None
        self.dateEnd = None
        self.select = None
        self.file_name = None
        self.append_file = None

        self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")

```

Here is the call graph for this function:


Core.ConstructionMonitor Init.__CreateFrame
Core.UtahRealEstateInit. __CreateFrame
AuthUtil.AuthUtil. __CreateFrame
API_Calls.Initializer.initializer. __CreateFrame
Core.UtahRealEstateInit. __init__
AuthUtil.AuthUtil. __ShowGui
AuthUtil.AuthUtil. __SetValues
Core.ConstructionMonitor Init.__ShowGui
Core.ConstructionMonitor Init.__SetValues
Core.UtahRealEstateInit. __ShowGui
Core.UtahRealEstateInit. __SetValues
API_Calls.Initializer.initializer. __ShowGui

3.14.3 Member Function Documentation

3.14.3.1 __CreateFrame()

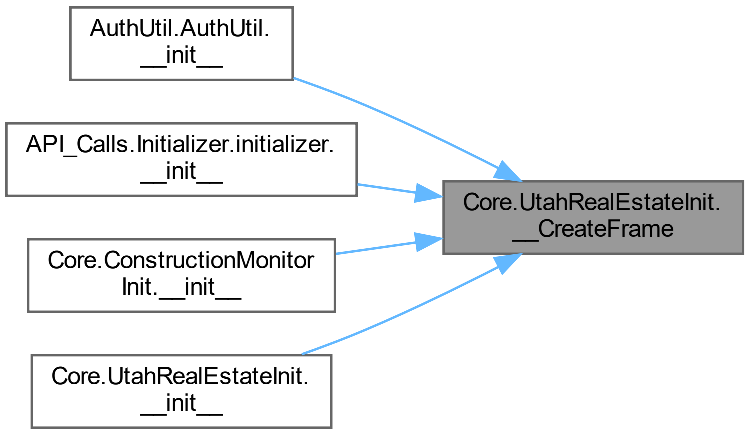
```
def Core.UtahRealEstateInit.__CreateFrame ( )  [static], [private]
```

```
The __CreateFrame function creates the GUI layout for the application.
The function returns a list of lists that contains all the elements to be displayed in the window.
Each element is defined by its type and any additional parameters needed to define it.

Args:

Returns:
A list of lists, which is used to create the gui

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 93 of file UtahRealEstate/Core.py.

```
00093     def __CreateFrame():
00094         """
00095     The __CreateFrame function creates the GUI layout for the application.
00096         The function returns a list of lists that contains all the elements to be displayed in the window.
00097         Each element is defined by its type and any additional parameters needed to define it.
00098 
00099     Args:
00100 
00101     Returns:
00102         A list of lists, which is used to create the gui
00103 
00104     Doc Author:
00105         Willem van der Schans, Trelent AI
00106     """
00107         sg.theme('Default1')
00108 
00109         line00 = [sg.HSeparator()]
00110 
00111         line0 = [sg.Image(ImageLoader("logo.png")),
00112                  sg.Push(),
00113                  sg.Text("Utah Real Estate Utility", font=("Helvetica", 12, "bold"),
      justification="center"),
00114                  sg.Push(),
00115                  sg.Push()]
00116 
00117         line1 = [sg.HSeparator()]
00118 
00119         line2 = [sg.Text("MLS Status : ", size=(15, None), justification="Right"),
00120                  sg.DropDown(default_value="Active", values=["Active", "Closed"], key="-status-",
      size=(31, 1))]
00121 
00122         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00123                  sg.DropDown(default_value="Listing Date", values=["Listing Date", "Modification Date",
      "Close Date"],
00124                              key="-type-", size=(31, 1))]
00125 
00126         line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00127                  sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"),
      key="-DateStart-",
00128                           disabled=False, size=(20, 1)),
00129                  sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
      target="-DateStart-")]
00130 
00131         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00132                  sg.Input(default_text=(date.today().strftime("%Y-%m-%d")), key="-DateEnd-",
      disabled=False,
00133                           size=(20, 1)),
00134                  sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-end_date-',
      target="-DateEnd-")]
00135 
00136         line7 = [sg.HSeparator()]
00137 
00138         line8 = [sg.Push(),
00139                  sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
00140                  sg.Push()]
00141 
00142         line9 = [sg.HSeparator()]
00143 
00144         line10 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
00145                   sg.Input(default_text="", key="-AppendingFile-", disabled=True,
00146                            size=(20, 1)),
00147                   sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
00148                                 target="-AppendingFile-")]
00149 
00150         line11 = [sg.HSeparator()]
00151 
00152         line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00153 
00154         layout = [line00, line0, line1, line2, line3, line4, line5, line7, line8, line9, line10, line11,
00155                   line12]
00156 
00157         return layout
00158 
```

Here is the caller graph for this function:


AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
Core.UtahRealEstateInit.__CreateFrame

3.14.3.2 __SetValues()

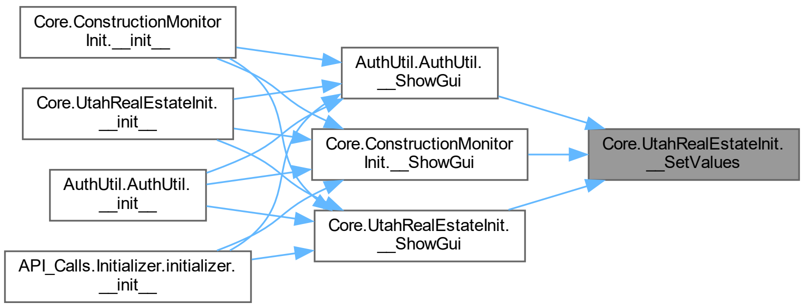
```
def Core.UtahRealEstateInit.__SetValues (
              self,
              values )  [private]
```

```
The __SetValues function is used to set the values of the variables that are used in the
__GetData function. The values are passed from a dictionary called 'values' which is created
by parsing through an XML file using ElementTree. This function also sets default values for
some of these variables if they were not specified in the XML file.

Args:
self: Represent the instance of the class
values: Pass the values from the gui to this function

Returns:
A dictionary with the following keys:

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 159 of file UtahRealEstate/Core.py.

```
    def __SetValues(self, values):

        """
    The __SetValues function is used to set the values of the variables that are used in the
       __GetData function. The values are passed from a dictionary called 'values' which is created
       by parsing through an XML file using ElementTree. This function also sets default values for
       some of these variables if they were not specified in the XML file.

    Args:
        self: Represent the instance of the class
        values: Pass the values from the gui to this function

    Returns:
        A dictionary with the following keys:

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = values["-status-"]

        self.ListedOrModified = values["-type-"]

        if values["-DateStart-"] != "":
            self.dateStart = values["-DateStart-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-DateEnd-"] != "":
            self.dateEnd = values["-DateEnd-"]
        else:
            self.dateEnd = (date.today()).strftime("%Y-%m-%d")

        self.select = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None


```

Here is the caller graph for this function:


Core.ConstructionMonitorInit.__init__
Core.UtahRealEstateInit.__init__
AuthUtil.AuthUtil.__init__
API_Calls.Initializer.initializer.__init__
AuthUtil.AuthUtil.__ShowGui
Core.ConstructionMonitorInit.__ShowGui
Core.UtahRealEstateInit.__ShowGui
Core.UtahRealEstateInit.__SetValues

3.14.3.3 __ShowGui()

```
def Core.UtahRealEstateInit.__ShowGui (
                self,
                layout,
                text )  [private]
```

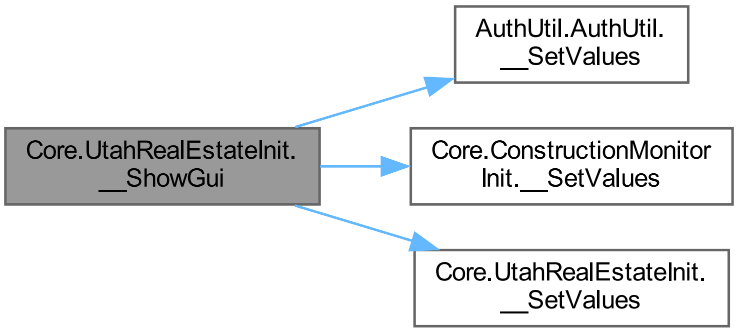
The __ShowGui function is a helper function that creates the GUI window and displays it to the user. It takes in two parameters: layout, which is a list of lists containing all the elements for each row; and text, which is a string containing what will be displayed as the title of the window. The __ShowGui

```
method then uses these parameters to create an instance of sg.Window with all its attributes set accordingly.

Args:
self: Refer to the current class instance
layout: Pass the layout of the window to be created
text: Set the title of the window

Returns:
A dictionary of values

Doc Author:
Willem van der Schans, Trelent AI
```

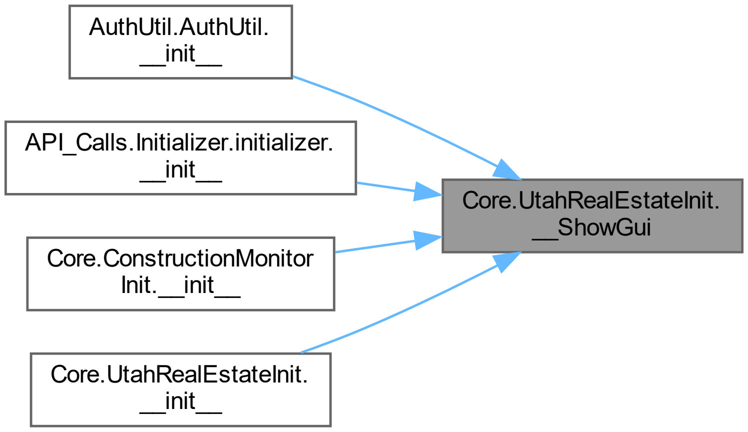
Definition at line 53 of file UtahRealEstate/Core.py.

```
00053     def __ShowGui(self, layout, text):
00054 
00055         """
00056     The __ShowGui function is a helper function that creates the GUI window and displays it to the user.
00057     It takes in two parameters: layout, which is a list of lists containing all the elements for each row;
00058     and text, which is a string containing what will be displayed as the title of the window. The
      __ShowGui
00059     method then uses these parameters to create an instance of sg.Window with all its attributes set
      accordingly.
00060 
00061     Args:
00062         self: Refer to the current class instance
00063         layout: Pass the layout of the window to be created
00064         text: Set the title of the window
00065 
00066     Returns:
00067         A dictionary of values
00068 
00069     Doc Author:
00070         Willem van der Schans, Trelent AI
00071     """
00072         window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00073                            finalize=True,
00074                            icon=ImageLoader("taskbar_icon.ico"))
00075 
00076         while True:
00077             event, values = window.read()
00078 
00079             if event == "Submit":
00080                 try:
00081                     self.__SetValues(values)
00082                     break
00083                 except Exception as e:
00084                     print(e)
00085                     RESTError(993)
00086                     raise SystemExit(993)
00087             elif event == sg.WIN_CLOSED or event == "Quit":
00088                 break
00089 
00090         window.close()
00091 
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.14.4 Member Data Documentation

3.14.4.1 append_file

```
Core.UtahRealEstateInit.append_file
```

Definition at line 49 of file UtahRealEstate/Core.py.

3.14.4.2 dateEnd

```
Core.UtahRealEstateInit.dateEnd
```

Definition at line 46 of file UtahRealEstate/Core.py.

3.14.4.3 dateStart

```
Core.UtahRealEstateInit.dateStart
```

Definition at line 45 of file UtahRealEstate/Core.py.

3.14.4.4 file_name

```
Core.UtahRealEstateInit.file_name
```

Definition at line 48 of file UtahRealEstate/Core.py.

3.14.4.5 ListedOrModified

```
Core.UtahRealEstateInit.ListedOrModified
```

Definition at line 44 of file UtahRealEstate/Core.py.

3.14.4.6 select

```
Core.UtahRealEstateInit.select
```

Definition at line 47 of file UtahRealEstate/Core.py.

3.14.4.7 StandardStatus

```
Core.UtahRealEstateInit.StandardStatus
```

Definition at line 43 of file UtahRealEstate/Core.py.

The documentation for this class was generated from the following file:

- UtahRealEstate/Core.py

3.15 Core.UtahRealEstateMain Class Reference

Public Member Functions

- def __init__ (self, siteClass)
- def mainFunc (self)

Public Attributes

- dataframe
- keyPath
- filePath
- key

Private Member Functions

- def __ParameterCreator (self)
- def __getCount (self)
- def __getCountUI (self)

Private Attributes

- __batches
- __siteClass
- __headerDict
- __parameterString
- __appendFile
- __dateStart
- __dateEnd
- __restDomain
- __record_val

3.15.1 Detailed Description

Definition at line 199 of file UtahRealEstate/Core.py.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 __init__()

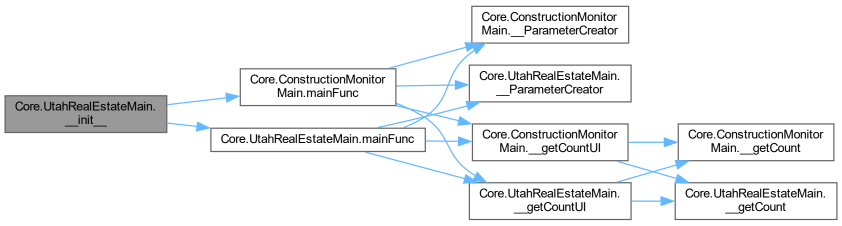
```
def Core.UtahRealEstateMain.__init__ (
              self,
              siteClass )
```

```
The __init__ function is the first function that runs when an object of this class is created.
It sets up all the variables and functions needed for this class to work properly.

Args:
self: Represent the instance of the class
siteClass: Determine which site to pull data from

Returns:
Nothing

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 201 of file UtahRealEstate/Core.py.

```
    def __init__(self, siteClass):

        """
    The __init__ function is the first function that runs when an object of this class is created.
    It sets up all the variables and functions needed for this class to work properly.

    Args:
        self: Represent the instance of the class
        siteClass: Determine which site to pull data from

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__batches = 0
        self.__siteClass = siteClass
        self.__headerDict = None
        self.__parameterString = ""
        self.__appendFile = None
        self.__dateStart = None
        self.__dateEnd = None
        self.__restDomain = settings.settingURERestDomain
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
            "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            "Security").joinpath("auth.json")
        self.key = None
        self.__record_val = None

        try:
            self.mainFunc()
        except KeyError as e:
            # This allows for user cancellation of the program using the quit button
            if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

```

Here is the call graph for this function:



3.15.3 Member Function Documentation

3.15.3.1 __getCount()

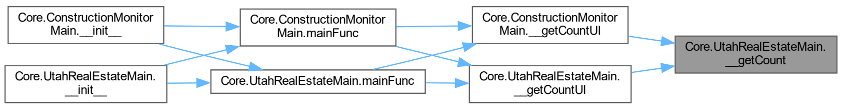
```
def Core.UtahRealEstateMain.__getCount (
                self ) [private]
```

```
The __getCount function is used to determine the number of records that will be returned by the query.
This function is called when a user calls the count() method on a ReST object. The __getCount function uses
the $count parameter in OData to return only an integer value representing how many records would be returned
by the query.

Args:
self: Represent the instance of the class

Returns:
The number of records in the data set

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 366 of file UtahRealEstate/Core.py.

```
    def __getCount(self):
        """
    The __getCount function is used to determine the number of records that will be returned by the query.
    This function is called when a user calls the count() method on a ReST object. The __getCount function
 uses
    the $count parameter in OData to return only an integer value representing how many records would be
 returned
    by the query.

    Args:
        self: Represent the instance of the class

    Returns:
        The number of records in the data set

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __count_resp = None

        try:
            __count_resp = requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
                                        headers=self.__headerDict)

        except requests.exceptions.Timeout as e:
            print(e)
            RESTError(790)
            raise SystemExit(790)
        except requests.exceptions.TooManyRedirects as e:
            print(e)
            RESTError(791)
            raise SystemExit(791)
        except requests.exceptions.MissingSchema as e:
            print(e)
            RESTError(1101)
        except requests.exceptions.RequestException as e:
            print(e)
            RESTError(405)
            raise SystemExit(405)

        self.__record_val = int(__count_resp.json()["@odata.count"])

```

Here is the caller graph for this function:


Core.ConstructionMonitorMain.__init__
Core.ConstructionMonitorMain.mainFunc
Core.ConstructionMonitorMain.__getCountUI
Core.UtahRealEstateMain.__getCount
Core.UtahRealEstateMain.__init__
Core.UtahRealEstateMain.mainFunc
Core.UtahRealEstateMain.__getCountUI

3.15.3.2 __getCountUI()

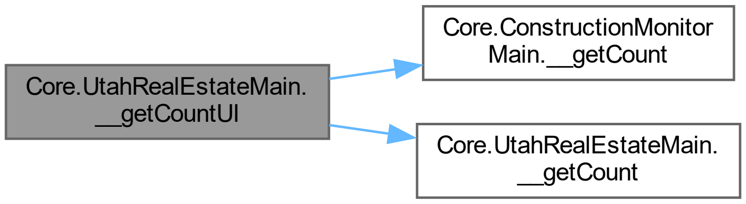
```
def Core.UtahRealEstateMain.__getCountUI (
              self )  [private]
```

```
The __getCountUI function is a wrapper for the __getCount function.
It creates a progress window and updates it while the __getCount function runs.
The purpose of this is to keep the GUI responsive while running long processes.

Args:
self: Represent the instance of the class

Returns:
A popupwrapped object

Doc Author:
Willem van der Schans, Trelent AI
```

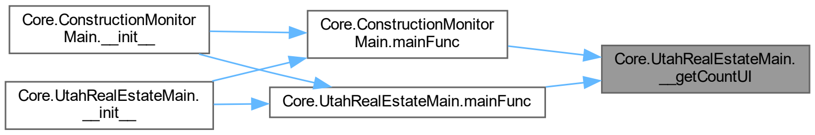
Definition at line 406 of file UtahRealEstate/Core.py.

```
    def __getCountUI(self):

        """
    The __getCountUI function is a wrapper for the __getCount function.
    It creates a progress window and updates it while the __getCount function runs.
    The purpose of this is to keep the GUI responsive while running long processes.

    Args:
        self: Represent the instance of the class

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__getCount,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()
```

Here is the call graph for this function:



Here is the caller graph for this function:



3.15.3.3 __ParameterCreator()

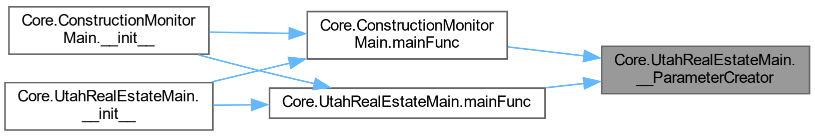
```
def Core.UtahRealEstateMain.__ParameterCreator (
              self )  [private]
```

```
The __ParameterCreator function is used to create the filter string for the ReST API call.
The function takes in a siteClass object and extracts all of its parameters into a dictionary.
It then creates an appropriate filter string based on those parameters.

Args:
self: Bind the object to the class

Returns:
A string to be used as the parameter in the api call

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 325 of file UtahRealEstate/Core.py.

```
00325     def __ParameterCreator(self):
00326         """
00327     The __ParameterCreator function is used to create the filter string for the ReST API call.
00328     The function takes in a siteClass object and extracts all of its parameters into a dictionary.
00329     It then creates an appropriate filter string based on those parameters.
00330 
00331     Args:
00332         self: Bind the object to the class
00333 
00334     Returns:
00335         A string to be used as the parameter in the api call
00336 
00337     Doc Author:
00338         Willem van der Schans, Trelent AI
00339     """
00340         filter_string = ""
00341 
00342         __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
00343                          not key.startswith('__') and not callable(key)}
00344 
00345         self.__appendFile = __Source_dict["append_file"]
00346         __Source_dict.pop("append_file")
00347 
00348         temp_dict = copy.copy(__Source_dict)
00349         for key, value in temp_dict.items():
00350             if value is None:
00351                 __Source_dict.pop(key)
00352             else:
00353                 pass
00354 
00355         if __Source_dict["ListedOrModified"] == "Listing Date":
00356             filter_string =
      f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContractDate%20le%20{__Source_dict['dateEn
00357         elif __Source_dict["ListedOrModified"] == "Modification Date":
00358             filter_string =
      f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20and%20ModificationTimestamp%20le%20{__Sour
00359         elif __Source_dict["ListedOrModified"] == "Close Date":
00360             filter_string =
      f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{__Source_dict['dateEnd']}"
00361 
00362         filter_string = filter_string +
      f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['StandardStatus']}'"
00363 
00364         self.__parameterString = filter_string
00365 
```

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.ConstructionMonitor Main.mainFunc
Core.UtahRealEstateMain. __ParameterCreator
Core.UtahRealEstateMain. __init__
Core.UtahRealEstateMain.mainFunc

3.15.3.4 mainFunc()

```
def Core.UtahRealEstateMain.mainFunc (
              self )
```

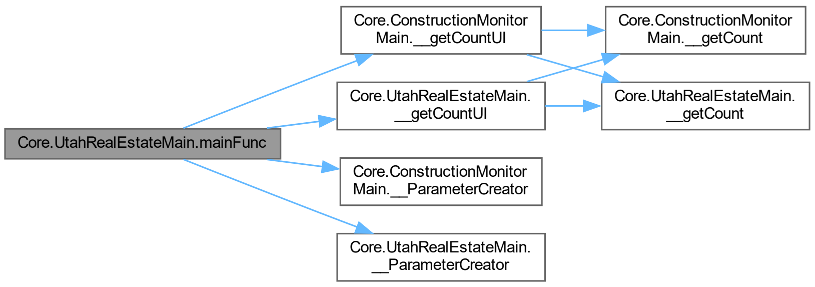
```
The mainFunc function is the main function of this module. It will be called by the GUI when a user clicks on
the &quot;Run&quot; button in the GUI. The mainFunc function should contain all of your code for running your pro
should return a dataframe that contains all the data you want to display in your final report.

Args:
self: Reference the object itself

Returns:
A dataframe

Doc Author:
Willem van der Schans, Trelent AI
```

Definition at line 248 of file UtahRealEstate/Core.py.

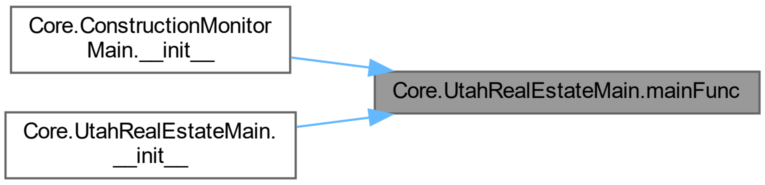
```
    def mainFunc(self):

        """
    The mainFunc function is the main function of this module. It will be called by the GUI when a user
clicks on
    the &quot;Run&quot; button in the GUI. The mainFunc function should contain all of your code for
running your program, and it
    should return a dataframe that contains all the data you want to display in your final report.

    Args:
        self: Reference the object itself

    Returns:
        A dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        passFlag = False

        while not passFlag:
            if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
                try:
                    f = open(self.keyPath, "rb")
                    key = f.readline()
                    f.close()
                    f = open(self.filePath, "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
                    self.__headerDict = {authDict["ure"]["parameter"]: authkey}
                    passFlag = True
                except Exception as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ParameterCreator()

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param String =
{self.__parameterString}")
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain =
{self.__restDomain}")

```

```
        self.__getCountUI()

        if self.__record_val is None:
            self.__record_val = 0

        self.__batches = BatchCalculator(self.__record_val, None)

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches = {self.__batches} | Rows {self.__record_val}")

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            eventReturn = BatchInputGui(self.__batches, self.__record_val)
            if eventReturn == "Continue":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches sent to server")
                BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                                  ParameterDict=self.__parameterString,
                                                  HeaderDict=self.__headerDict,
                                                  BatchesNum=self.__batches,
                                                  Type="utah_real_estate")
                BatchGuiObject.BatchGuiShow()
                self.dataframe = BatchGuiObject.dataframe
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) - startTime).total_seconds()))}")
                FileSaver("ure", self.dataframe, self.__appendFile)
            else:
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches canceled by user")
        else:
            RESTError(994)
            raise SystemExit(994)

```

Here is the call graph for this function:


Core.UtahRealEstateMain.mainFunc
Core.ConstructionMonitorMain.__getCountUI
Core.UtahRealEstateMain.__getCountUI
Core.ConstructionMonitorMain.__ParameterCreator
Core.UtahRealEstateMain.__ParameterCreator
Core.ConstructionMonitorMain.__getCount
Core.UtahRealEstateMain.__getCount

Here is the caller graph for this function:


Core.ConstructionMonitor Main.__init__
Core.UtahRealEstateMain. __init__
Core.UtahRealEstateMain.mainFunc

3.15.4 Member Data Documentation

3.15.4.1 __appendFile

`Core.UtahRealEstateMain.__appendFile [private]`

Definition at line 222 of file UtahRealEstate/Core.py.

3.15.4.2 __batches

`Core.UtahRealEstateMain.__batches [private]`

Definition at line 218 of file UtahRealEstate/Core.py.

3.15.4.3 __dateEnd

`Core.UtahRealEstateMain.__dateEnd [private]`

Definition at line 224 of file UtahRealEstate/Core.py.

3.15.4.4 __dateStart

```
Core.UtahRealEstateMain.__dateStart  [private]
```

Definition at line 223 of file UtahRealEstate/Core.py.

3.15.4.5 __headerDict

```
Core.UtahRealEstateMain.__headerDict  [private]
```

Definition at line 220 of file UtahRealEstate/Core.py.

3.15.4.6 __parameterString

```
Core.UtahRealEstateMain.__parameterString  [private]
```

Definition at line 221 of file UtahRealEstate/Core.py.

3.15.4.7 __record_val

```
Core.UtahRealEstateMain.__record_val  [private]
```

Definition at line 231 of file UtahRealEstate/Core.py.

3.15.4.8 __restDomain

```
Core.UtahRealEstateMain.__restDomain  [private]
```

Definition at line 225 of file UtahRealEstate/Core.py.

3.15.4.9 __siteClass

```
Core.UtahRealEstateMain.__siteClass  [private]
```

Definition at line 219 of file UtahRealEstate/Core.py.

3.15.4.10 dataframe

```
Core.UtahRealEstateMain.dataframe
```

Definition at line 217 of file UtahRealEstate/Core.py.

3.15.4.11 filePath

```
Core.UtahRealEstateMain.filePath
```

Definition at line 228 of file UtahRealEstate/Core.py.

3.15.4.12 key

```
Core.UtahRealEstateMain.key
```

Definition at line 230 of file UtahRealEstate/Core.py.

3.15.4.13 keyPath

```
Core.UtahRealEstateMain.keyPath
```

Definition at line 226 of file UtahRealEstate/Core.py.

The documentation for this class was generated from the following file:

- UtahRealEstate/Core.py

Chapter 4

File Documentation

4.1 __init__.py

4.2 _main_.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


from Initializer import initializer

initializer()
```

4.3 AuthUtil.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


import ctypes
import datetime
import json
import os
from pathlib import Path

import PySimpleGUI as sg
from cryptography.fernet import Fernet

from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class AuthUtil:

    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the initial state of the object, which in this case means that it creates a new window and
displays it on screen.

    Args:
        self: Represent the instance of the class

    Returns:
        None

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.file_name = None
        self.append_file = None
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security'))
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath("Security")
        self.k = None
        self.keyFlag = True
        self.jsonDict = {}
        self.passFlagUre = False
        self.passFlagCm = False
        self.outcomeText = "Please input the plain text keys in the input boxes above \n " \
                           "Submitting will overwrite any old values in an unrecoverable manner."

        if os.path.exists(self.filePath):
            pass
        else:
            if os.path.exists(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData")):
                os.mkdir(self.filePath)
            else:
                os.mkdir(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData"))
                os.mkdir(self.filePath)

        if os.path.exists(self.keyPath):
            pass
        else:
            if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
                os.mkdir(self.keyPath)
            else:
                os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
                os.mkdir(self.keyPath)

        if os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
            try:
                f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                self.k = f.readline()
                f.close()
            except Exception as e:
                print(e)
                RESTError(402)
                raise SystemExit(402)
        else:
            self.k = Fernet.generate_key()
            f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
            f.write(self.k)
            f.close()

            try:
                os.remove(self.filePath.joinpath("auth.json"))
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py | Error = {e} | Error in removing auth.json file - This can be due to the file not existing. Continuing...")
                pass

            f = open(self.filePath.joinpath("auth.json"), "wb")
            f.close()
            self.keyFlag = False

        self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")

        try:
            ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), 2)
        except Exception as e:
            # Logging
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error when setting the key file as hidden. This is either a Permission error or Input Error. Continuing...")
            pass

    def __SetValues(self, values):

        """
    The __SetValues function is called when the user clicks on the &quot;OK&quot; button in the window.
    It takes a dictionary of values as an argument, and then uses those values to update
    the auth.json file with new keys for both Utah Real Estate and Construction Monitor.

    Args:
        self: Make the function a method of the class
        values: Store the values that are entered into the form

    Returns:
        A dictionary of the values entered by the user

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        ureCurrent = None
        cmCurrent = None
        keyFile = None
        self.popupFlag = False

        fernet = Fernet(self.k)

        try:
            f = open(self.filePath.joinpath("auth.json"), "r")
            keyFile = json.load(f)
            fileFlag = True
        except:
            fileFlag = False

        # Try initial decoding, if fails pass and write new keys and files
        if fileFlag:
            try:
                ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Utah Real Estate Key. Continuing but this should be resolved if URE functionality will be accessed")
                ureCurrent = None

            try:
                cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
            except Exception as e:
                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Authutil.py |Error = {e} | Error decoding Construction Monitor Key. Continuing but this should be resolved if CM functionality will be accessed")
                cmCurrent = None

        if values["-ureAuth-"] != "":
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
            self.passFlagUre = True
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"ure": {"parameter": "Authorization", "auth": fernet.encrypt(ureCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if values["-cmAuth-"] != "":
            if values["-cmAuth-"].startswith("Basic"):
                self.jsonDict.update(
                    {"cm": {"parameter": "Authorization",
                            "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
                self.passFlagCm = True
            else:
                PopupWrapped("Please make sure you provide a HTTP Basic Auth key for construction Monitor",
                             windowType="AuthError")
                self.popupFlag = True
                pass
        elif ureCurrent is not None:
            self.jsonDict.update(
                {"cm": {"parameter": "Authorization", "auth": fernet.encrypt(cmCurrent.encode()).decode()}})
            self.passFlagUre = True
        else:
            pass

        if not self.passFlagUre and not self.passFlagCm:
            PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction Monitor",
                         windowType="errorLarge")
        if self.passFlagCm and not self.passFlagUre:
            PopupWrapped("Please make sure you provide a key for Utah Real estate", windowType="errorLarge")
        if not self.passFlagCm and self.passFlagUre and not self.popupFlag:
            PopupWrapped("Please make sure you provide a key for Construction Monitor", windowType="errorLarge")
        if self.popupFlag:
            pass
        else:
            jsonOut = json.dumps(self.jsonDict, indent=4)
            f = open(self.filePath.joinpath("auth.json"), "w")
            f.write(jsonOut)

    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is a helper function that displays the GUI to the user.
    It takes in two arguments: layout and text. The layout argument is a list of lists,
    which contains all the elements that will be displayed on screen. The text argument
    is simply what will be displayed at the top of the window.

    Args:
        self: Represent the instance of the class
        layout: Pass the layout of the gui to be displayed
        text: Set the title of the window

    Returns:
        A window object
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while not self.passFlagUre or not self.passFlagCm:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                except Exception as e:
                    print(e)
                    RESTError(993)
                finally:
                    pass
            elif event == sg.WIN_CLOSED or event == "Quit":

                break
            else:
                pass

        window.close()

    def __CreateFrame(self):
        """
    The __CreateFrame function creates the GUI layout for the Authentication Utility.
    It is called by __init__ and returns a list of lists that contains all the elements
    that will be displayed in the window.

    Args:
        self: Access the class attributes and methods

    Returns:
        A list of lists

    Doc Author:
        Trelent
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"), justification="center"),
                sg.Push(),
                sg.Push()]

        line1 = [sg.HSeparator()]

        line2 = [sg.Push(),
                 sg.Text("Utah Real Estate API Key: ", justification="center"),
                 sg.Push()]

        line3 = [sg.Push(),
                 sg.Input(default_text="123", key="-ureAuth-", disabled=False,
                          size=(40, 1)),
                 sg.Push()]

        line4 = [sg.HSeparator()]

        line5 = [sg.Push(),
                 sg.Text("Construction Monitor HTTP BASIC Key: ", justification="center"),
                 sg.Push()]

        line6 = [sg.Push(),
                 sg.Input(default_text="Basic 123", key="-cmAuth-", disabled=False,
                          size=(40, 1)),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Push(),
                 sg.Text(self.outcomeText, justification="center"),
                 sg.Push()]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout
```

4.4 BatchProcessing.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


import datetime
import math
from datetime import date

import pandas as pd
import requests

from API_Calls.Functions.DataFunc.DataSupportFunctions import StringToList


def BatchCalculator(TotalRecords, Argument_Dict):
    """
The BatchCalculator function takes two arguments:
    1. TotalRecords - the total number of records in the database
    2. Argument_Dict - a dictionary containing all the arguments passed to this function by the user

Args:
    TotalRecords: Determine the number of batches that will be needed to complete the query
    Argument_Dict: Pass in the arguments that will be used to query the database

Returns:
    The total number of batches that will be made

Doc Author:
    Willem van der Schans, Trelent AI
"""
    try:
        document_limit = Argument_Dict["size"]
    except Exception as e:
        # Logging
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProcessing.py
|Error = {e} | Batch Calculator document limit overwritten to 200 from input")
        document_limit = 200

    return int(math.ceil(float(TotalRecords) / float(document_limit)))


class BatchProcessorConstructionMonitor:

    def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject):

        """
    The __init__ function is the constructor for a class. It is called when an object of that class
    is created, and it sets up the attributes of that object. In this case, we are setting up our
    object to have a dataframe attribute (which will be used to store all of our data), as well as
    attributes for each parameter in our ReST call.

    Args:
        self: Represent the instance of the class
        RestDomain: Specify the domain of the rest api
        NumBatches: Determine how many batches of data to retrieve
        ParameterDict: Pass in the parameters that will be used to make the api call
        HeaderDict: Pass the header dictionary from the main function to this class
        ColumnSelection: Determine which columns to pull from the api
        valueObject: Pass in the value object that is used to determine what values are returned

    Returns:
        An object of the class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterDict = ParameterDict
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.__columnSelection = ColumnSelection
        self.valueObject = valueObject
        self.__maxRequests = 10000
        self.__requestCount = math.ceil(self.__numBatches / (self.__maxRequests /
int(self.__parameterDict['size'])))
        self.__requestCalls = math.ceil(self.__maxRequests / int(self.__parameterDict['size']))
        self.__dateTracker = None

    def FuncSelector(self):
        """
    The FuncSelector function is a function that takes the valueObject and passes it to the
ConstructionMonitorProcessor function.
    The ConstructionMonitorProcessor function then uses this valueObject to determine which of its
functions should be called.

    Args:
        self: Represent the instance of the class

    Returns:
        The result of the constructionmonitorprocessor function

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.ConstructionMonitorProcessor(self.valueObject)

    def ConstructionMonitorProcessor(self, valueObject):
        """
    The ConstructionMonitorProcessor function will use requests to get data from
       ConstructionMontior.com's ReST API and store it into a pandas DataFrame object called __df (which
is local). This
       process will be repeated until all the data has been collected from ConstructionMonitor.com's ReST
API, at which point __df will contain all

    Args:
        self: Represent the instance of the object itself
        valueObject: Update the progress bar in the gui

    Returns:
        A dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
```

```
        __df = None
        for callNum in range(0, self.__requestCount):
            self.__parameterDict["from"] = 0

            if self.__requestCount > 1 and callNum != self.__requestCount - 1:
                __batchNum = self.__requestCalls
                if __df is None:
                    self.__dateTracker = str(date.today())
                else:
                    self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
            elif self.__requestCount == 1:
                __batchNum = self.__numBatches
                self.__dateTracker = str(date.today())
            else:
                __batchNum = self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size'])) - (
                        self.__requestCount - 1)
                self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')

            self.__parameterDict['dateEnd'] = self.__dateTracker

            for record in range(0, int(math.ceil(__batchNum))):
                if record != 0:
                    self.__parameterDict["from"] = record * int(self.__parameterDict["size"])

                response = requests.post(url=self.__restDomain,
                                         headers=self.__headerDict,
                                         json=self.__parameterDict)

                counter = 0
                try:
                    response = response.json()['hits']['hits']
                except KeyError as e:
                    # Logging
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProcessing.py |Error = {e} | Count Request Error Server Response: {response.json()} | Batch = {record} | Parameters = {self.__parameterDict} | Headers = {self.__headerDict}")
                    continue

                valueObject.setValue(valueObject.getValue() + 1)

                if record == 0 and callNum == 0:
                    __df = pd.json_normalize(response[counter]["_source"])
                    __df["id"] = response[counter]['_id']
                    __df["county"] = response[counter]["_source"]['county']['county_name']
                    counter += 1

                for i in range(counter, len(response)):
                    __tdf = pd.json_normalize(response[i]["_source"])
                    __tdf["id"] = response[i]['_id']
                    __tdf["county"] = response[i]["_source"]['county']['county_name']
                    __df = pd.concat([__df, __tdf], ignore_index=True)

        if self.__columnSelection is not None:
            __col_list = StringToList(self.__columnSelection)
            __col_list.append("id")
            __col_list.append("county")
        else:
            pass

        self.dataframe = __df
        valueObject.setValue(-999)


class BatchProcessorUtahRealEstate:

    def __init__(self, RestDomain, NumBatches, ParameterString, HeaderDict, valueObject):
        """
    The __init__ function is the constructor for a class. It is called when an object of that class
    is instantiated, and it sets up the attributes of that object. In this case, we are setting up
    the dataframe attribute to be None (which will be set later), and we are also setting up some
    other attributes which will help us make our API calls.

    Args:
        self: Represent the instance of the class
        RestDomain: Specify the domain of the rest api
        NumBatches: Determine how many batches of data to pull from the api
        ParameterString: Pass the parameters to the rest api
        HeaderDict: Pass in the header information for the api call
        valueObject: Create a dataframe from the json response

    Returns:
        The instance of the class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__numBatches = NumBatches
        self.__parameterString = ParameterString
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.valueObject = valueObject

    def FuncSelector(self):
        """
    The FuncSelector function is a function that takes the valueObject as an argument and then calls the
appropriate
        function based on what was selected in the dropdown menu.  The valueObject is passed to each of
these functions
        so that they can access all of its attributes.

    Args:
        self: Represent the instance of the class

    Returns:
        The function that is selected by the user

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.BatchProcessingUtahRealestateCom(self.valueObject)

    def BatchProcessingUtahRealestateCom(self, valueObject):
        """
    The BatchProcessingUtahRealestateCom function is a function that takes in the valueObject and uses it
to
       update the progress bar. It also takes in self, which contains all the necessary information for
this
       function to work properly. The BatchProcessingUtahRealestateCom function will then use requests to
get data from
       UtahRealestate.com's ReST API and store it into a pandas DataFrame object called __df (which is
local). This
       process will be repeated until all the data has been collected from UtahRealestate.com's ReST API,
at which point __df will contain all

    Args:
        self: Represent the instance of the class
        valueObject: Pass the value of a progress bar to the function

    Returns:
        A dataframe of the scraped data

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __df = pd.DataFrame()

        for batch in range(self.__numBatches):

            if batch == 0:
                response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
                                        headers=self.__headerDict)

                response_temp = response.json()
                __df = pd.json_normalize(response_temp, record_path=['value'])

            else:
                response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch
* 200}",
                                        headers=self.__headerDict)

                response_temp = response.json()
                response_temp = pd.json_normalize(response_temp, record_path=['value'])
                __df = pd.concat([__df, response_temp], ignore_index=True)

            valueObject.setValue(valueObject.getValue() + 1)

        self.dataframe = __df
        valueObject.setValue(-999)
```

4.5 DataSupportFunctions.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


def StringToList(string):
    """
The StringToList function takes a string and converts it into a list.
    The function is used to convert the input from the user into a list of strings, which can then be iterated through.

Args:
    string: Split the string into a list

Returns:
    A list of strings

Doc Author:
    Willem van der Schans, Trelent AI
"""
    listOut = list(string.split(","))
    return listOut
```

4.6 FileSaver.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime
import os
from pathlib import Path

import pandas as pd

from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class FileSaver:

    def __init__(self, method, outputDF, AppendingPath=None):
        """
    The __init__ function is called when the class is instantiated.
    It sets up the instance of the class, and defines all variables that will be used by other functions in this class.
    The __init__ function takes two arguments: self and method.  The first argument, self, refers to an instance of a
    class (in this case it's an instance of DataFrameSaver). The second argument, method refers to a string value that
    is passed into DataFrameSaver when it's instantiated.

    Args:
        self: Represent the instance of the class
        method: Determine which dataframe to append the new data to
        outputDF: Pass in the dataframe that will be saved to a csv file
        AppendingPath: Specify the path to an existing csv file that you want to append your dataframe to

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.docPath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            datetime.datetime.today().strftime('%m%d%Y'))
        self.data = outputDF
        self.dataAppending = None
        self.appendFlag = True
        self.fileName = f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
        self.uiFlag = True

        if method.lower() == "ure":
            self.primaryKey = "ListingKeyNumeric"
        elif method.lower() == "cm":
            self.primaryKey = "id"
        elif "realtor" in method.lower():
            self.primaryKey = None
            self.uiFlag = False
        elif method.lower() == "cfbp":
            self.primaryKey = None
            self.uiFlag = False
        else:
            raise ValueError("method input is invalid choice one of 4 options: URE, CM, Realtor, CFBP")

        if AppendingPath is None:
            self.appendFlag = False
        else:
            self.dataAppending = pd.read_csv(AppendingPath)

        if self.appendFlag:
            if self.primaryKey is not None:
                # Due to low_memory loading the columns are not typed properly,
                # since we are comparing this will be an issue since we need to do type comparisons,
                # so here we coerce the types of the primary keys to numeric.
                # If another primary key is ever chosen make sure to core to the right data type.
                self.dataAppending[self.primaryKey] = pd.to_numeric(self.dataAppending[self.primaryKey])
                self.data[self.primaryKey] = pd.to_numeric(self.data[self.primaryKey])

                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(subset=[self.primaryKey],
                                                                                              keep="last")
            else:
                self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(keep="last")
        else:
            self.outputFrame = self.data

        if os.path.exists(self.docPath):
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
        else:
            os.mkdir(self.docPath)
            self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)

        if self.uiFlag:
            if self.appendFlag:
                PopupWrapped(text=f"File Appended and Saved to {self.docPath.joinpath(self.fileName)}",
                             windowType="savedLarge")

                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | {method} API request Completed | File Appended and Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0")
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Appending Statistics | Method: {method} | Appending file rows: {self.dataAppending.shape[0]}, Total Rows: {(self.dataAppending.shape[0] + self.data.shape[0])}, Duplicates Dropped {(self.dataAppending.shape[0] + self.data.shape[0])-self.outputFrame.shape[0]}")
            else:
                PopupWrapped(text=f"File Saved to {self.docPath.joinpath(self.fileName)}", windowType="savedLarge")

                # Logging
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | {method} API request Completed | File Saved to {self.docPath.joinpath(self.fileName)} | Exit Code 0")
        else:
            pass

    def getPath(self):
        """
    The getPath function returns the path to the file.
        It is a string, and it joins the docPath with the fileName.

    Args:
        self: Represent the instance of the class

    Returns:
        The path to the file

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        return str(self.docPath.joinpath(self.fileName))
```

4.7 Settings.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


# Setting NameSpace for maintenance

class settings:
    #   Version Checker
    #       Update accordingly using semantic versioning: https://semver.org/
    settingVersion = "1.2.0"
    #       Update in conjunction with settingDownloadSourceLink
    settingGithubApiUrl = "https://api.github.com/repos/Kydoimos97/GardnerApiUtility/releases/latest"

    #   PopUpWrapped
    #       Singular Reference free to change
    settingGenerationToolLink = 'https://www.debugbear.com/basic-auth-header-generator'
    #       Update in conjunction with settingDownloadSourceLink
    settingDownloadSourceLink = 'https://github.com/Kydoimos97/GardnerApiUtility/releases/latest'

    # CFBP Source
    #       This link downloads csv's immediately so minimal change is likely required in the source code
    settingCFBPLink = "https://ffiec.cfpb.gov/v2/data-browser-api/view/csv?"

    # ConstructionMonitor Source
    #       Check the REST call and data parser when updating this
    settingCMRestDomain = "https://api.constructionmonitor.com/v2/powersearch/?"

    # Realtor.Com Source
    #       Updating This link likely requires a rewrite of the html parser
    settingRealtorLink = "https://www.realtor.com/research/data/"

    # UtahRealEstate Source
    #       API links are generated with hard references so updating this link requires a large code
rewrite
    settingURERestDomain = "https://resoapi.utahrealestate.com/reso/odata/Property?"
```

4.8 versionChecker.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/
import requests

from API_Calls.Functions.DataFunc.Settings import settings
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


def versionChecker():
    """
The versionChecker function is used to check if the current version of the program is up-to-date.
It does this by comparing the latest release on GitHub.
If they are not equal, it will pop up a window telling you that there's an update available.

Args:

Returns:
    A popup window with the current version and latest version

Doc Author:
    Willem van der Schans, Trelent AI
"""
    # Todo Gitlab Update
    current_version = settings.settingVersion
    # Todo Gitlab Update
    response = requests.get(settings.settingGithubApiUrl)
    latest_version = response.json()['name']
    text_string = f"A different version is tagged as latest release \n \n" \
                  f"Running version: {current_version}\n" \
                  f"Latest version: {latest_version}"
    print(text_string)

    if current_version != latest_version:
        PopupWrapped(text_string, windowType="versionWindow")
```

4.9 ErrorPopup.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


def ErrorPopup(textString):
    """
The ErrorPopup function is used to display a popup window with an error message.
It takes one argument, textString, which is the string that will be displayed in the popup window.
The function also opens up the log folder upon program exit.

Args:
    textString: Display the error message

Returns:
    Nothing, but it does print an error message to the console

Doc Author:
    Willem van der Schans, Trelent AI
"""
    PopupWrapped(
        f"ERROR @ {textString} \n"
        f"Log folder will be opened upon program exit",
        windowType="FatalErrorLarge")
```

4.10 ErrorPrint.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


import datetime


def RESTErrorPrint(response):
    """
The RESTErrorPrint function is used to print the response from a ReST API call.
If the response is an integer, it will be printed as-is. If it’s not an integer,
it will be converted to text and then printed.

Args:
    response: Print the response from a rest api call

Returns:
    The response text

Doc Author:
    Willem van der Schans, Trelent AI
"""
    if isinstance(response, int):
        print(f"{datetime.datetime.today().strftime(’%m-%d-%Y %H:%M:%S.%f’)[:-3]} | Resource Response:
{response}")
    else:
        response_txt = response.text
        print(f"{datetime.datetime.today().strftime(’%m-%d-%Y %H:%M:%S.%f’)[:-3]} | Resource Response:
{response_txt}")
```

4.11 Logger.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


import datetime
import os
import sys
from pathlib import Path

```

```python

def logger():
    """
The logger function creates a log file in the user’s AppData directory.
The function will create the directory if it does not exist.
The function will also delete the oldest file when 100 logs have been saved to prevent bloat.

Args:

Returns:
    A file path to the log file that was created

Doc Author:
    Willem van der Schans, Trelent AI
"""
    dir_path = Path(os.path.expandvars(r’%APPDATA%\GardnerUtil\Logs’))
    if os.path.exists(dir_path):
        pass
    else:
        if os.path.exists(Path(os.path.expandvars(r’%APPDATA%\GardnerUtil’))):
            os.mkdir(dir_path)
        else:
            os.mkdir(Path(os.path.expandvars(r’%APPDATA%\GardnerUtil’)))
            os.mkdir(dir_path)

    filePath = Path(os.path.expandvars(r’%APPDATA%\GardnerUtil\Logs’)).joinpath(
        f"{datetime.datetime.today().strftime(’%m%d%Y_%H%M%S’)}.log")
    sys.stdout = open(filePath, ’w’)
    sys.stderr = sys.stdin = sys.stdout

    def sorted_ls(path):
        """
    The sorted_ls function takes a path as an argument and returns the files in that directory sorted by modification time.

    Args:
        path: Specify the directory to be sorted

    Returns:
        A list of files in a directory sorted by modification time

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        mtime = lambda f: os.stat(os.path.join(path, f)).st_mtime
        return list(sorted(os.listdir(path), key=mtime))

    del_list = sorted_ls(dir_path)[0:(len(sorted_ls(dir_path)) - 100)]
    for file in del_list:
        os.remove(dir_path.joinpath(file))
        print(f"{datetime.datetime.today().strftime(’%m-%d-%Y %H:%M:%S.%f’)[:-3]} | Log file {file} deleted")
```

4.12 RESTError.py

```python
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/


import datetime

from API_Calls.Functions.ErrorFunc.ErrorPopup import ErrorPopup
from API_Calls.Functions.ErrorFunc.ErrorPrint import RESTErrorPrint


def RESTError(response):
    """
The RESTError function is a function that checks the status codes.
If it is 200, then everything went well and nothing happens. If it isn’t 200, then an error message will be printed to
the console with information about what happened (i.e., if there was an authentication error or if the resource wasn’t found).
The function also raises an exception and opens an error popup for easy debugging.

Args:
    response: Print out the response from the server

```

```
Returns:
    A text string

Doc Author:
    Trelent
"""
    if isinstance(response, int):
        status_code = response
    else:
        status_code = response.status_code

    if status_code == 200:
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Api Request completed successfully"
        print(textString)
        pass
    elif status_code == 301:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Endpoint redirection; check domain name and endpoint name"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 400:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Bad Request; check input arguments"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 401:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Authentication Error: No keys found"
        ErrorPopup(textString)
        raise PermissionError(textString)
    elif status_code == 402:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Authentication Error: Cannot access decryption Key in
%appdata%/roaming/GardnerUtil/security"
        ErrorPopup(textString)
        raise PermissionError(textString)
    elif status_code == 403:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Access Error: the resource you are trying to access is forbidden"
        ErrorPopup(textString)
        raise PermissionError(textString)
    elif status_code == 404:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Resource not found: the resource you are trying to access does not exist on the server"
        ErrorPopup(textString)
        raise NameError(textString)
    elif status_code == 405:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Method is not valid, request rejected by server"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 408:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Requests timeout by server"
        ErrorPopup(textString)
        raise TimeoutError(textString)
    elif status_code == 503:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | The resource is not ready for the get request"
        ErrorPopup(textString)
        raise SystemError(textString)
    elif status_code == 701:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Error in coercing icon to bits (Imageloader.py)"
        ErrorPopup(textString)
        raise TypeError(textString)
    elif status_code == 801:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code =
{status_code} | Resource Error, HTML cannot be parsed the website's HTML source might be changed"
```

```
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 790:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Requests timeout within requests"
        ErrorPopup(textString)
        raise TimeoutError(textString)
    elif status_code == 791:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Too many redirects, Bad url"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 990:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No password input"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 991:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No username input"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 992:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | No authentication input (Basic or User/PW)"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 993:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Submission Error: input values could not be coerced to arguments"
        ErrorPopup(textString)
        print(ValueError(textString))
    elif status_code == 994:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Submission Error: server returned no documents"
        ErrorPopup(textString)
        raise ValueError(textString)
    elif status_code == 1000:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Catastrophic Error"
        ErrorPopup(textString)
        raise SystemError(textString)
    elif status_code == 1001:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | Main Function Error Break"
        raise SystemError(textString)
    elif status_code == 1100:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | User has cancelled the program execution"
        raise KeyboardInterrupt(textString)
    elif status_code == 1101:
        RESTErrorPrint(response)
        textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | User returned to main menu using the exit button"
        print(textString)
    else:
        RESTErrorPrint(response)
        raise Exception(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Status Code = {status_code} | An unknown exception occurred")
```

4.13 BatchGui.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on http://www.apache.org/licenses/

import PySimpleGUI as sg

```

```
from API_Calls.Functions.Gui.ImageLoader import ImageLoader


def BatchInputGui(batches, documentCount=None):
    """
The BatchInputGui function is a simple GUI that displays the number of batches and pages
that will be requested. It also gives the user an option to cancel or continue with their request.


Args:
    batches: Determine how many batches will be run
    documentCount: Determine how many documents will be retrieved

Returns:
    The event, which is the button that was pressed

Doc Author:
    Willem van der Schans, Trelent AI
"""
    event = None
    if documentCount is None:
        __text1 = f"This request will run {batches}"
    else:
        __text1 = f"This request will run {batches} batches and will retrieve {documentCount} rows"

    __text2 = "Press Continue to start request"

    __Line1 = [sg.Push(),
               sg.Text(__text1, justification="center"),
               sg.Push()]

    __Line2 = [sg.Push(),
               sg.Text(__text2, justification="center"),
               sg.Push()]

    __Line3 = [sg.Push(),
               sg.Ok("Continue"),
               sg.Cancel(),
               sg.Push()]

    window = sg.Window("Popup", [__Line1, __Line2, __Line3],
                       modal=True,
                       keep_on_top=True,
                       disable_close=True,
                       icon=ImageLoader("taskbar_icon.ico"))

    while True:
        event, values = window.read()
        if event == "Continue":
            break
        elif event == sg.WIN_CLOSED or event == "Cancel":
            break

    window.close()

    return event


def confirmDialog():
    """
The confirmDialog function is a simple confirmation dialog that asks the user if they want to continue
with the request.
The function takes no arguments and returns the button event to allow for process confirmation.

Args:

Returns:
    The event that was triggered,

Doc Author:
    Willem van der Schans, Trelent AI
"""
    event = None
    __text1 = f"This request can take multiple minutes to complete"
    __text2 = "Press Continue to start the request"

    __Line1 = [sg.Push(),
               sg.Text(__text1, justification="center"),
               sg.Push()]

    __Line2 = [sg.Push(),
               sg.Text(__text2, justification="center"),
               sg.Push()]

    __Line3 = [sg.Push(),
               sg.Ok("Continue"),
               sg.Cancel(),
               sg.Push()]

    window = sg.Window("Popup", [__Line1, __Line2, __Line3],
                       modal=True,
                       keep_on_top=True,
                       disable_close=True,
                       icon=ImageLoader("taskbar_icon.ico"))

    while True:
        event, values = window.read()
        if event == "Continue":
            break
        elif event == sg.WIN_CLOSED or event == "Cancel":
            break

    window.close()

    return event
```

4.14 BatchProgressGUI.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/

import datetime
import threading
import time

import PySimpleGUI as sg

from API_Calls.Functions.DataFunc.BatchProcessing import BatchProcessorConstructionMonitor,
BatchProcessorUtahRealEstate
from API_Calls.Functions.Gui.DataTransfer import DataTransfer
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped

counter = 1


class BatchProgressGUI:

    def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None):

        """
    The __init__ function is the first function that gets called when an object of this class is created.
    It initializes all the variables and sets up a layout for the GUI. It also creates a window to display
    the dataframe in.

    Args:
        self: Represent the instance of the class
        BatchesNum: Determine the number of batches that will be created
        RestDomain: Specify the domain of the rest api
        ParameterDict: Pass the parameters of the request to the class
        HeaderDict: Store the headers of the dataframe
        Type: Determine the type of dataframe that is being created
        ColumnSelection: Select the columns to be displayed in the gui

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__parameterDict = ParameterDict
        self.__restDomain = RestDomain
        self.__headerDict = HeaderDict
        self.__columnSelection = ColumnSelection
        self.__type = Type
        self.dataframe = None

        self.__layout = None
        self.__batches = BatchesNum
        self.__window = None
        self.__batch_counter = 0

    def BatchGuiShow(self):
        """
    The BatchGuiShow function is called by the BatchGui function. It creates a progress bar layout and then calls the createGui function to create a GUI for batch processing.

    Args:
        self: Represent the instance of the class

    Returns:
        The __type of the batchgui class

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.CreateProgressLayout()
        self.createGui(self.__type)

    def CreateProgressLayout(self):

        """
    The CreateProgressLayout function creates the layout for the progress window.
        The function takes in self as a parameter and returns nothing.

        Parameters:
            self (object): The object that is calling this function.

    Args:
        self: Access the class variables and methods

    Returns:
        A list of lists

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        __Line1 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center", key="--progress_text--"),
                   sg.Push()]

        __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center", key="--timer--"),
                   sg.Text(font=("Helvetica", 10), justification="center", key="--time_est--"), sg.Push()]

        __Line3 = [
            sg.ProgressBar(max_value=self.__batches, bar_color=("#920303", "#C9c8c8"), orientation='h', size=(30, 20),
                           key='--progress_bar--')]


        layout = [__Line1, __Line2, __Line3]

        self.__layout = layout

    def createGui(self, Sourcetype):

        """
    The createGui function is the main function that creates the GUI.
    It takes in a type parameter which determines what kind of batch processor to use.
    The createGui function then sets up all the variables and objects needed for
    the program to run, including: window, start_time, update_text, valueObj (DataTransfer),
    processorObject (BatchProcessorConstructionMonitor or BatchProcessorUtahRealestate),
    and threading objects for TimeUpdater and ValueChecker functions. The createGui function also starts these threads.

    Args:
        self: Access the object itself
        Sourcetype: Determine which batch processor to use

    Returns:
        The dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__window = sg.Window('Progress', self.__layout, finalize=True, icon=ImageLoader("taskbar_icon.ico"))
```

```python

        start_time = datetime.datetime.now().replace(microsecond=0)
        update_text = f"Batch {0} completed"
        self.__window['--progress_text--'].update(update_text)
        self.__window['--progress_bar--'].update(0)
        self.__window['--time_est--'].update("Est time needed 00:00:00")

        valueObj = DataTransfer()
        valueObj.setValue(0)

        if Sourcetype == "construction_monitor":

            processorObject = BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
                                                                NumBatches=self.__batches,
                                                                ParameterDict=self.__parameterDict,
                                                                HeaderDict=self.__headerDict,
                                                                ColumnSelection=self.__columnSelection,
                                                                valueObject=valueObj)
        elif Sourcetype == "utah_real_estate":
            processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
                                                           NumBatches=self.__batches,
                                                           ParameterString=self.__parameterDict,
                                                           HeaderDict=self.__headerDict,
                                                           valueObject=valueObj)

        threading.Thread(target=self.TimeUpdater,
                         args=(start_time,),
                         daemon=True).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | TimeUpdater Thread Successfully Started")

        batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
                                           daemon=False)
        batchFuncThread.start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchFunc Thread Successfully Started")
        threading.Thread(target=self.ValueChecker,
                         args=(valueObj,),
                         daemon=False).start()
        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | ValueChecker Thread Successfully Started")

        while True:

            self.ProgressUpdater(valueObj)

            if valueObj.getValue() == -999:
                break

            window, event, values = sg.read_all_windows()
            if event.startswith('update'):
                __key_to_update = event[len('update'):]
                window[__key_to_update].update(values[event])
                window.refresh()
                pass

            if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
                break

            time.sleep(0.1)

        self.dataframe = processorObject.dataframe
        self.__window.close()

        PopupWrapped(text="Api Request Completed", windowType="notice")

    def ProgressUpdater(self, valueObj):
        """
    The ProgressUpdater function is a callback function that updates the progress bar and text
    in the GUI. It takes in one argument, which is an object containing information about the
    current batch number. The ProgressUpdater function then checks if this value has changed from
    the last time it was called (i.e., if we are on a new batch). If so, it updates both the progress
    bar and text with this new information.

    Args:
        self: Make the progressupdater function an instance method
        valueObj: Get the current value of the batch counter

    Returns:
        The value of the batch counter

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        if valueObj.getValue() != self.__batch_counter:
            self.__batch_counter = valueObj.getValue()

            __update_text = f"Batch {self.__batch_counter}/{self.__batches} completed"

            self.__window.write_event_value('update--progress_bar--', self.__batch_counter)
            self.__window.write_event_value('update--progress_text--', __update_text)
        else:
            pass

    def TimeUpdater(self, start_time):

        """
    The TimeUpdater function is a thread that updates the time elapsed and estimated time needed to complete
    the current batch. It does this by reading the start_time variable passed in, getting the current time,
    calculating how much time has passed since start_time was set and then updating a timer string with that value.
    It then calculates an estimation of how long it will take to finish all batches based on how many batches have been completed so far.

    Args:
        self: Make the function a method of the class
        start_time: Get the time when the function is called

    Returns:
        A string that is updated every 0

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            if self.__batch_counter < self.__batches:

                __current_time = datetime.datetime.now().replace(microsecond=0)

                __passed_time = __current_time - start_time

                __timer_string = f"Time Elapsed {__passed_time}"

                try:
                    self.__window.write_event_value('update--timer--', __timer_string)
                except AttributeError as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | BatchProgressGUI.py | Error = {e} | Timer string attribute error, this is okay if the display looks good, this exception omits fatal crashes due to an aesthetic error")
                    break

                __passed_time = __passed_time.total_seconds()

                try:
                    __time_est = datetime.timedelta(
                        seconds=(__passed_time * (self.__batches / self.__batch_counter) - __passed_time)).seconds
                except:
                    __time_est = datetime.timedelta(
                        seconds=(__passed_time * self.__batches - __passed_time)).seconds

                __time_est = time.strftime('%H:%M:%S', time.gmtime(__time_est))

                __end_string = f"Est time needed {__time_est}"
                self.__window.write_event_value('update--time_est--', __end_string)
            else:
                __end_string = f"Est time needed 00:00:00"
                self.__window.write_event_value('update--time_est--', __end_string)
            time.sleep(0.25)

    def ValueChecker(self, ObjectVal):
        """
    The ValueChecker function is a thread that checks the value of an object.
        It will check if the value has changed, and if it has, it will return True.
        If not, then it returns False.

    Args:
        self: Represent the instance of the class
        ObjectVal: Get the value of the object

    Returns:
        True if the value of the object has changed, and false if it hasn’t

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            time.sleep(0.3)
            if self.__batch_counter != ObjectVal.getValue():
                self.__batch_counter = ObjectVal.getValue()
                return True
            else:
                return False
```

4.15 DataTransfer.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


class DataTransfer:

    def __init__(self):
        """
    The __init__ function is called when the class is instantiated.
    It sets the initial value of self.__value to 0.

    Args:
        self: Represent the instance of the class

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__value = 0

    def setValue(self, value):
        """
    The setValue function sets the value of the object.


    Args:
        self: Represent the instance of the class
        value: Set the value of the instance variable __value

    Returns:
        The value that was passed to it

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__value = value

    def getValue(self):
        """
    The getValue function returns the value of the private variable __value.
    This is a getter function that allows access to this private variable.

    Args:
        self: Represent the instance of the class

    Returns:
        The value of the instance variable

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        return self.__value

    def whileValue(self):
        """
    The whileValue function is a function that will run the getValue function until it is told to stop.
    This allows for the program to constantly be checking for new values from the sensor.

    Args:
        self: Refer to the current instance of the class

    Returns:
        The value of the input

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        while True:
            self.getValue()
```

4.16 ImageLoader.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


import base64
import os
from io import BytesIO
from os.path import join, normpath

from PIL import Image


def ImageLoader(file):
    """
The ImageLoader function takes in a file name and returns the image as a base64 encoded string.
This is used to send images to the API for processing.

Args:
    file: Specify the image file to be loaded

Returns:
    A base64 encoded image string

Doc Author:
    Willem van der Schans, Trelent AI
"""
    try:
        __path = normpath(join(str(os.getcwd().split("API_Calls", 1)[0]), "API_Calls"))
        __path = normpath(join(__path, "External Files"))
        __path = normpath(join(__path, "Images"))
        __path = join(__path, file).replace("\\", "/")

        image = Image.open(__path)

        __buff = BytesIO()

        image.save(__buff, format="png")

        img_str = base64.b64encode(__buff.getvalue())

        return img_str
    except Exception as e:
        # We cannot log this error like other errors due to circular imports
        raise e
```

4.17 PopupWrapped.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/
import datetime
import os
import threading
import time
import webbrowser
from pathlib import Path

import PySimpleGUI as sg

from API_Calls.Functions.DataFunc.Settings import settings
from API_Calls.Functions.Gui.ImageLoader import ImageLoader


class PopupWrapped():

    def __init__(self, text="", windowType="notice", error=None):
        """
    The __init__ function is the first function that gets called when an object of this class is created.
    It sets up all the variables and creates a window for us to use.
    Args:
        self: Represent the instance of the class
        text: Set the text of the window
        windowType: Determine what type of window to create
        error: Display the error message in the window
    Returns:
        Nothing
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__text = text
        self.__type = windowType
        self.__error = error
        self.__layout = []
        self.__windowObj = None
        self.__thread = None
        self.__counter = 0
        self.__docpath = None
        self.__errorFlag = False

        try:
            if "File Appended and Saved to " in self.__text:
                self.__docpath = str(self.__text[27:])
            elif "File Saved to " in self.__text:
                self.__docpath = str(self.__text[14:])
            else:
                pass
        except Exception as e:
            if self.__type == "savedLarge":
                print(
                    f"{datetime.datetime.today().strftime(’%m-%d-%Y %H:%M:%S.%f’)[:-3]} | PopupWrapped.py | Error = {e} | Error creating self.__docpath open file button not available")
                self.__errorFlag = True
            else:
                pass

        self.__createWindow()

    def __createLayout(self):
        """
    The __createLayout function is used to create the layout of the window.
    The function takes class variables and returns a window layout.
    It uses a series of if statements to determine what type of window it is, then creates a layout based on that information.
    Args:
        self: Refer to the current instance of a class
    Returns:
        A list of lists
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme(’Default1’)
        __Line1 = None
        __Line2 = None

        if self.__type == "notice":
            __Line1 = [sg.Push(),
                       sg.Text(u’\u2713’, font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "noticeLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u’\u2713’, font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "savedLarge":
            if self.__errorFlag:
                __Line1 = [sg.Push(),
                           sg.Text(u’\u2713’, font=("Helvetica", 20, "bold"), justification="center"),
                           sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
                __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
            else:
                __Line1 = [sg.Push(),
                           sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
                           sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
                __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "errorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "FatalErrorLarge":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(self.__text, justification="left", key="-textField-"), sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "error":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}: {self.__error}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "AuthError":
            __Line1 = [sg.Push(),
                       sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
                       sg.Text(f"{self.__text}", justification="center", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool [Web Browser]"),
                       sg.Ok(button_text="Return", focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "versionWindow":
            __Line1 = [sg.Push(),
                       sg.Text(f"{self.__text}", justification="left", key="-textField-"),
                       sg.Push()]
            __Line2 = [sg.Push(), sg.Button(button_text="Download"),
                       sg.Ok(button_text="Continue", focus=True, size=(10, 1)), sg.Push()]
        elif self.__type == "progress":
            __Line1 = [sg.Push(),
                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]

        if self.__type == "progress":
            self.__layout = [__Line1, ]
        else:
            self.__layout = [__Line1, __Line2]

    def __createWindow(self):
        """
    The __createWindow function is used to create the window object that will be displayed.
    The function takes class variables and a window object. The function first calls __createLayout, which creates the layout for the window based on what type of message it is (error, notice, progress). Then it uses PySimpleGUI's Window class to create a new window with that layout and some other parameters such as title and icon. If this is not a progress bar or permanent message then we start a timer loop that waits until either 100 iterations have passed or an event has been triggered (such as clicking &quot;Ok&quot; or closing the window). Once one of these events occurs
    Args:
        self: Reference the instance of the class
    Returns:
        A window object
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__createLayout()

        if self.__type == "progress":
            self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 50))
        elif self.__type == "noticeLarge":
            self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "savedLarge":
            self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=False,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "errorLarge":
            self.__windowObj = sg.Window(title="Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "FatalErrorLarge":
            self.__windowObj = sg.Window(title="Fatal Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "AuthError":
            self.__windowObj = sg.Window(title="Authentication Error", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        elif self.__type == "versionWindow":
            self.__windowObj = sg.Window(title="Update Notice", layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"))
        else:
            self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout, finalize=True,
                                         modal=True,
                                         keep_on_top=True,
                                         disable_close=False,
                                         icon=ImageLoader("taskbar_icon.ico"),
                                         size=(290, 80))

        if self.__type != "progress" or self.__type.startswith("perm"):
            timer = 0
            while timer < 100:
                event, values = self.__windowObj.read()
                if event == "Ok" or event == sg.WIN_CLOSED or event == "Return" or event == "Continue":
                    break
                elif event == "Open Generation Tool [Web Browser]":
                    webbrowser.open(settings.settingGenerationToolLink, new=2, autoraise=True)
                    pass
                elif event == "Open File":
                    threadFile = threading.Thread(target=self.openFile,
                                                  daemon=False)
                    threadFile.start()
                    time.sleep(3)
                    break
                elif event == "Download":
                    # Todo Gitlab Update
                    webbrowser.open(settings.settingDownloadSourceLink, new=2,
                                    autoraise=True)
                    pass
                time.sleep(0.1)

            if self.__type == "FatalErrorLarge":
                try:
                    os.system(
                        f"start {Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}")
                except Exception as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | PopupWrapped.py | Error = {e} | Log Folder not found please search manually for %APPDATA%\Roaming\GardnerUtil\Logs\n")

            self.__windowObj.close()

    def stopWindow(self):
        """
    The stopWindow function is used to close the window object that was created in the startWindow function.
    This is done by calling the close() method on self.__windowObj, which will cause it to be destroyed.
    Args:
        self: Represent the instance of the class
    Returns:
        The window object
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__windowObj.close()

    def textUpdate(self, sleep=0.5):
        """
    The textUpdate function is a function that updates the text in the text field.
    It does this by adding dots to the end of it, and then removing them. This creates
    a loading effect for when something is being processed.
    Args:
        self: Refer to the object itself
        sleep: Control the speed of the text update
    Returns:
        A string that is the current text of the text field
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__counter += 1
        if self.__counter == 4:
            self.__counter = 1
        newString = ""
        if self.__type == "notice":
            pass
        elif self.__type == "error":
            pass
        elif self.__type == "progress":
            newString = f"{self.__text}{'.' * self.__counter}"
        self.__windowObj.write_event_value('update-textField-', newString)

        time.sleep(sleep)

    def windowPush(self):

        """
    The windowPush function is used to update the values of a window object.
        The function takes in an event and values from the window object, then checks if the event starts
with 'update'.
        If it does, it will take everything after 'update' as a key for updating that specific value.
        It will then update that value using its key and refresh the window.
    Args:
        self: Reference the object that is calling the function
    Returns:
        A tuple containing the event and values
    Doc Author:
        Willem van der Schans, Trelent AI
    """
        event, values = self.__windowObj.read()

        if event.startswith('update'):
            __key_to_update = event[len('update'):]
            self.__windowObj[__key_to_update].update(values[event])
            self.__windowObj.refresh()

    def openFile(self):
        """
    The openFile function opens the file that is associated with the
        document object.  It does this by calling os.system and passing it
        self.__docpath as an argument.

    Args:
        self: Represent the instance of the object itself

    Returns:
        The filepath of the document

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        os.system(self.__docpath)
```

4.18 Initializer.py

```
#   This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/


import datetime
import os
```

```
from pathlib import Path

import PySimpleGUI as sg

from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
from API_Calls.Functions.DataFunc.versionChecker import versionChecker
from API_Calls.Functions.ErrorFunc.Logger import logger
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
from API_Calls.Sources.CFBP.Core import CFBP
from API_Calls.Sources.ConstructionMonitor.Core import ConstructionMonitorInit, \
    ConstructionMonitorMain
from API_Calls.Sources.Realtor.Core import realtorCom
from API_Calls.Sources.UtahRealEstate.Core import UtahRealEstateMain, UtahRealEstateInit


class initializer:

    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the logging, calls the __ShowGui function to create and display
    the GUI, and then calls __CreateFrame to create a frame for displaying widgets.


    Args:
        self: Represent the instance of the class

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.classObj = None

        logger()

        print("\n\n-------------Initiate Program---------------------\n\n")

        self.__ShowGui(self.__CreateFrame(), "Data Tool")

        print("\n\n-------------Closing Program---------------------\n\n")

    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is the main function that displays the GUI.
    It takes two arguments: layout and text. Layout is a list of lists, each containing a tuple with three elements:
        1) The type of element to be displayed (e.g., &quot;Text&quot;, &quot;InputText&quot;, etc.)
        2) A dictionary containing any additional parameters for that element (e.g., size, default value, etc.)
        3) An optional key name for the element (used in event handling). If no key name is provided then one will be generated automatically by PySimpleGUIQt based on its position in the layout list

    Args:
        self: Represent the instance of the class
        layout: Pass the layout of the window to be created
        text: Set the title of the window

    Returns:
        A window object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        # Todo Gitlab Update
        versionChecker()

        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Construction Monitor":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Initiating Construction Monitor API Call-----------------")
                ConstructionMonitorMain(ConstructionMonitorInit())
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Closing Construction Monitor API Call---------------------\n")
            elif event == "Utah Real Estate":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Initiating Utah Real Estate API Call-----------------")
                UtahRealEstateMain(UtahRealEstateInit())
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Closing Utah Real Estate API Call---------------------\n")
            elif event == "Realtor.Com":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Initiating Realtor.com API Call-----------------")
                realtorCom()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Closing Realtor.com API Call---------------------\n")
            elif event == "CFPB Mortgage":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Initiating ffiec.cfpb API Call-----------------")
                CFBP()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Closing ffiec.cfpb API Call---------------------\n")
            elif event == "Authorization Utility":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Initiating Authorization Utility-----------------")
                AuthUtil()
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Closing Authorization Utility---------------------\n")
            elif event == "Open Data Folder":
                print(
                    f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
-------------Data Folder Opened-----------------")
                try:
                    os.system(f"start
{Path(os.path.expanduser('~/Documents')).joinpath('GardnerUtilData')}")
                except:
                    try:
                        os.system(f"start {Path(os.path.expanduser('~/Documents'))}")
                    except Exception as e:
                        print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
Initializer.py | Error = {e} | Documents folder not found")
                        PopupWrapped(
                            text="Documents folder not found. Please create a Windows recognized documents
folder",
                            windowType="errorLarge")

            elif event in ('Exit', None):
                try:
                    break
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
Initializer.py | Error = {e} | Error on program exit, for logging purposes only.")
                    break
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

    def __CreateFrame(self):

        """
    The __CreateFrame function is a helper function that creates the layout for the main window.
    It returns a list of lists, which is then passed to sg.Window() as its layout parameter.

    Args:
        self: Represent the instance of the class

    Returns:
        A list of lists, which is then passed to the sg

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line0 = [sg.HSeparator()]

        line1 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line3 = [sg.HSeparator()]

        line4 = [sg.Push(),
                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)), sg.Push(),
                  sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]

        line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(),
                  sg.Button("CFPB Mortgage", size=(20, None)),
                  sg.Push()]]

        line8 = [sg.HSeparator()]

        line9 = [sg.Push(),
                 sg.Text("Utilities", font=("Helvetica", 10, "bold"), justification="center"),
                 sg.Push()]

        line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
                   sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]

        line11 = [sg.HSeparator()]

        layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10, line11]

        return layout
```

4.19 CFBP/Core.py

```
import datetime
import threading
import time

import pandas as pd
import requests

from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.DataFunc.Settings import settings
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import confirmDialog
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class CFBP:

    def __init__(self, state_arg=None, year_arg=None):
        """
    The __init__ function is called when the class is instantiated.
    Its job is to initialize the object with some default values, and do any other setup that might be
necessary.
    The __init__ function can take arguments, but it doesn't have to.

    Args:
        self: Represent the instance of the class
        state_arg: Set the state_arg attribute of the class
        year_arg: Set the year of data to be retrieved

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.state_arg = state_arg
        self.year_arg = year_arg
        self.uiString = None
        self.link = None

        eventReturn = confirmDialog()
        if eventReturn == "Continue":
            startTime = datetime.datetime.now().replace(microsecond=0)
            self.__showUi()
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | API Link =
{self.link}")
            F = FileSaver("cfbp", pd.read_csv(self.link, low_memory=False))
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Data retrieved with
in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")

            self.uiString = (
                f"ffiec.cfpb.gov (Mortgage API) request Completed \n {self.year_arg} data retrieved \n
Data Saved at {F.getPath()}")

            PopupWrapped(text=self.uiString, windowType="noticeLarge")
        else:
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled
Request")
            pass

    def __showUi(self):

        """
    The __showUi function is a function that creates a progress bar window.
    The __showUi function takes class variables and returns a windowobj.


    Args:
        self: Represent the instance of the class

    Returns:
        The uiobj variable

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Cenus Request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__dataGetter,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()

    def __dataGetter(self):
        """
    The __dataGetter function is a private function that gets the data from the CFPB API.
    It takes no arguments, but uses self.state_arg and self.year_arg to create a URL for the API call.

    Args:
        self: Represent the instance of the class

    Returns:
        A response object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        arg_dict_bu = locals()

        link = settings.settingCFBPLink

        if self.state_arg is None:
            self.state_arg = "UT"
        else:
            pass

        if self.year_arg is None:
            self.year_arg = str(datetime.date.today().year - 1)
        else:
            pass

        passFlag = False

        while not passFlag:

            self.link = link + f"states={self.state_arg}" + f"&years={self.year_arg}"

            response = requests.get(self.link)

            if response.status_code == 400:
                self.year_arg = int(self.year_arg) - 1

            else:
                passFlag = True

        RESTError(response)
        raise SystemExit(0)
```

4.20 ConstructionMonitor/Core.py

```
import copy
import datetime
import json
import os
import threading
import time
from datetime import date, timedelta
from pathlib import Path

import PySimpleGUI as sg
import requests
from cryptography.fernet import Fernet

from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.DataFunc.Settings import settings
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import BatchInputGui
from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class ConstructionMonitorInit:

    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the variables that will be used by other functions in this class.


    Args:
        self: Represent the instance of the class

    Returns:
        None

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.size = None
        self.SourceInclude = None
        self.dateStart = None
        self.dateEnd = None
        self.rest_domain = None
        self.auth_key = None
        self.ui_flag = None
        self.append_file = None

        passFlag = False

        while not passFlag:
            if os.path.isfile(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                    "3v45wfvw45wvc4f35.av3ra3rvavcr3w")) and os.path.isfile(
                Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                    "Security").joinpath("auth.json")):
                try:
                    f = open(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
                        "3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
                    key = f.readline()
                    f.close()
                    f = open(Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
                        "Security").joinpath("auth.json"), "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    self.auth_key = fernet.decrypt(authDict["cm"]["auth"]).decode()
                    passFlag = True
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | ConstructionMonitor/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ShowGui(self.__CreateFrame(), "Construction Monitor Utility")

    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is the main function that creates and displays the GUI.
    It takes in a layout, which is a list of lists containing all the elements to be displayed on screen.
    The text parameter specifies what title should appear at the top of the window.

    Args:
        self: Refer to the current instance of a class
        layout: Determine what the gui will look like
        text: Set the title of the window

    Returns:
        A dictionary of values

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(933)
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

    @staticmethod
    def __CreateFrame():

        """
    The __CreateFrame function creates the GUI layout for the application.
        The function returns a list of lists that contains all the elements to be displayed in the GUI window.
        This is done by creating each line as a list and then appending it to another list which will contain all lines.

    Args:

    Returns:
        The layout for the gui

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"),
justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"),
key="-Cal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]

        line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-",
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
target="-EndCal-")]

        line5 = [sg.HSeparator()]

        line6 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push()]

        line7 = [sg.HSeparator()]

        line8 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
                          size=(20, 1)),
                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                               target="-AppendingFile-")]

        line9 = [sg.HSeparator()]

        line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]

        return layout

    def __SetValues(self, values):

        """
    The __SetValues function is used to set the values of the variables that are used in the __GetData
function.
    The __SetValues function takes a dictionary as an argument, and then sets each variable based on what
is passed into
    the dictionary. The keys for this dictionary are defined by the user when they create their own
instance of this class.

    Args:
        self: Represent the instance of the class
        values: Pass in the values from the ui

    Returns:
        A dictionary of values

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.size = 1000

        if values["-Cal-"] != "":
            self.dateStart = values["-Cal-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-EndCal-"] != "":
            self.dateEnd = values["-EndCal-"]
        else:
            self.dateEnd = date.today().strftime("%Y-%m-%d")

        self.rest_domain = settings.settingCMRestDomain

        self.SourceInclude = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None

        self.ui_flag = True


class ConstructionMonitorMain:

    def __init__(self, siteClass):

        """
    The __init__ function is the first function that runs when an object of this class is created.
    It sets up all the variables and functions needed for this class to run properly.


    Args:
        self: Represent the instance of the class
        siteClass: Identify the site that is being used

    Returns:
        Nothing

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__siteClass = siteClass
        self.__restDomain = None
        self.__headerDict = None
        self.__columnSelection = None
        self.__appendFile = None

        self.__parameterDict = {}
        self.__search_id = None
        self.__record_val = 0
        self.__batches = 0

        self.__ui_flag = None

        self.dataframe = None

        try:
            self.mainFunc()
        except SystemError as e:
            if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Cooerced SystemError in ConstructionMonitorMain class")
                pass
        except AttributeError as e:
            # This allows for user cancellation of the program using the quit button
            if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Error {e}")
                pass
            elif e is not None:
                print(
                    f"ConstructionMonitor/Core.py | Error = {e} | Authentication Error | Please update keys in AuthUtil")
                RESTError(401)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

    def mainFunc(self):
        """
    The mainFunc function is the main function of this module. It will be called by the GUI or CLI to execute
    the code in this module. The mainFunc function will first create a parameter dictionary using the __ParameterCreator
    method, then it will get a count of all records that match its parameters using the __getCountUI method, and then
    it will calculate how many batches are needed to retrieve all records with those parameters using BatchCalculator.
    After that it asks if you want to continue with retrieving data from Salesforce (if running in GUI mode). Then it shows
    a progress bar for each

    Args:
        self: Refer to the current object

    Returns:
        The dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__ParameterCreator()

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param Dict =
{self.__parameterDict}")
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain =
{self.__restDomain}")

        self.__getCountUI()

        self.__batches = BatchCalculator(self.__record_val, self.__parameterDict)

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches =
{self.__batches} | Rows {self.__record_val}")

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            eventReturn = BatchInputGui(self.__batches, self.__record_val)
            if eventReturn == "Continue":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for
{self.__batches} batches sent to server")
                BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                                  ParameterDict=self.__parameterDict,
                                                  HeaderDict=self.__headerDict,
                                                  ColumnSelection=self.__columnSelection,
                                                  BatchesNum=self.__batches,
                                                  Type="construction_monitor")
                BatchGuiObject.BatchGuiShow()
                self.dataframe = BatchGuiObject.dataframe
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Dataframe
retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in
{time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
                FileSaver("cm", self.dataframe, self.__appendFile)
            else:
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for
{self.__batches} batches canceled by user")
        else:
            RESTError(994)
            raise SystemExit(994)

    def __ParameterCreator(self):
        """
    The __ParameterCreator function is used to create the parameter dictionary that will be passed into
the
        __Request function. The function takes in a siteClass object and extracts all of its attributes,
except for
        those that start with '__' or are callable. It then creates a dictionary from these attributes and
stores it as
        self.__parameterDict.

    Args:
        self: Make the function a method of the class

    Returns:
        A dictionary of parameters and a list of non parameter variables

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                         not key.startswith('__') and not callable(key)}

        self.__restDomain = __Source_dict["rest_domain"]
        __Source_dict.pop("rest_domain")
        self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
        __Source_dict.pop("auth_key")
        self.__columnSelection = __Source_dict["SourceInclude"]
        __Source_dict.pop("SourceInclude")
        self.__ui_flag = __Source_dict["ui_flag"]
        __Source_dict.pop("ui_flag")
        self.__appendFile = __Source_dict["append_file"]
        __Source_dict.pop("append_file")

        temp_dict = copy.copy(__Source_dict)
        for key, value in temp_dict.items():
            if value is None:
                __Source_dict.pop(key)
            else:
                pass

        self.__parameterDict = copy.copy(__Source_dict)

    def __getCount(self):
        """
    The __getCount function is used to get the total number of records that are returned from a query.
    This function is called by the __init__ function and sets the self.__record_val variable with this value.

    Args:
        self: Represent the instance of the class

    Returns:
        The total number of records in the database

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __count_resp = None

        try:

            __temp_param_dict = copy.copy(self.__parameterDict)

            __count_resp = requests.post(url=self.__restDomain,
                                         headers=self.__headerDict,
                                         json=__temp_param_dict)

        except requests.exceptions.Timeout as e:
            print(e)
            RESTError(790)
            raise SystemExit(790)
        except requests.exceptions.TooManyRedirects as e:
            print(e)
            RESTError(791)
            raise SystemExit(791)
        except requests.exceptions.MissingSchema as e:
            print(e)
            RESTError(1101)
        except requests.exceptions.RequestException as e:
            print(e)
            RESTError(405)
            raise SystemExit(405)

        __count_resp = __count_resp.json()

        self.__record_val = __count_resp["hits"]["total"]["value"]

        del __count_resp, __temp_param_dict

    def __getCountUI(self):

        """
    The __getCountUI function is a wrapper for the __getCount function.
    It allows the user to run __getCount in a separate thread, so that they can continue working while it runs.
    The function will display a progress bar and update with text as it progresses through its tasks.

    Args:
        self: Access the class variables and methods

    Returns:
        The count of the number of records in the database

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        if self.__ui_flag:
            uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

            threadGui = threading.Thread(target=self.__getCount,
                                         daemon=False)
            threadGui.start()

            while threadGui.is_alive():
                uiObj.textUpdate()
                uiObj.windowPush()
            else:
                uiObj.stopWindow()

        else:
            self.__getCount()
```

4.21 Realtor/Core.py

```
import datetime
import threading
import time

import pandas as pd
import requests
from bs4 import *

from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.DataFunc.Settings import settings
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import confirmDialog
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class realtorCom:

    def __init__(self):
        """
    The __init__ function is called when the class is instantiated.
    It sets up the initial state of an object, and it's where you put code that needs to run before
anything else in your class.

    Args:
        self: Represent the instance of the class

    Returns:
        A new object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.__page_html = None
        self.__update_date = None
        self.__last_date = None
        self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
        self.__linkDict = {}
        self.dfState = None
        self.dfCounty = None
        self.dfZip = None
        self.uiString = "Files Saved to \n"

        eventReturn = confirmDialog()
        if eventReturn == "Continue":
            page_html = requests.get(settings.settingRealtorLink).text
            self.__page_html = BeautifulSoup(page_html, "html.parser")
            startTime = datetime.datetime.now().replace(microsecond=0)
            self.__linkGetter()
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Link Dictionary =
{self.__idDict}")
            self.__showUi()
            PopupWrapped(text=self.uiString, windowType="noticeLarge")
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Data retrieved with
in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) -
startTime).total_seconds()))}")
        else:
            print(
                f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | User Canceled Request")
            pass

    def __showUi(self):

        """
    The __showUi function is a helper function that creates and displays the progress window.
    It also starts the dataUpdater thread, which will update the progress bar as it runs.


    Args:
        self: Represent the instance of the class

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__dataUpdater,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()

    def __linkGetter(self):

        """
    The __linkGetter function is a private function that takes the idDict dictionary and adds
    a link to each entry in the dictionary. The link is used to access historical data for each
    scope symbol.

    Args:
        self: Refer to the object itself

    Returns:
        A dictionary of all the links to the history pages

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        for key, value in self.__idDict.items():
            for row in self.__page_html.find_all("div", {"class": "monthly"}):
                try:
                    for nestedRow in row.find_all("a"):
                        if "History" in str(nestedRow.get("href")) and key in str(nestedRow.get("href")):
                            self.__idDict[key] = {"id": value, "link": nestedRow.get("href")}
                except Exception as e:
                    print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Realtor/Core.py | Error = {e} | Error while getting document links for realtor.com")
                    RESTError(801)
                    raise SystemExit(801)

    def __dataUpdater(self):

        """
    The __dataUpdater function is a private function that updates the dataframes for each of the three
        types of realtor data. It takes class variables and return the path to the saved file. The function first creates an empty
        dictionary called tempdf, then iterates through each key in self.__idDict (which contains all three ids).
        For each key, it reads in a csv file from the link associated with that id and saves it to tempdf as a pandas
        DataFrame object. Then, depending on which type of realtor data we are dealing with (State/County/Zip), we save


    Args:
        self: Access the attributes and methods of the class

    Returns:
        The path of the saved file

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        for key, value in self.__idDict.items():
            tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)

            if key == "State":
                self.dfState = tempdf
            elif key == "County":
                self.dfCounty = tempdf
            elif key == "Zip":
                self.dfZip = tempdf

            FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
            self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()} \n"
```

4.22 UtahRealEstate/Core.py

```
import copy
import datetime
import json
import os
import threading
import time
from datetime import date, timedelta
from pathlib import Path

import PySimpleGUI as sg
import requests
from cryptography.fernet import Fernet

from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
from API_Calls.Functions.DataFunc.FileSaver import FileSaver
from API_Calls.Functions.DataFunc.Settings import settings
from API_Calls.Functions.ErrorFunc.RESTError import RESTError
from API_Calls.Functions.Gui.BatchGui import BatchInputGui
from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
from API_Calls.Functions.Gui.ImageLoader import ImageLoader
from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped


class UtahRealEstateInit:

    def __init__(self):

        """
    The __init__ function is called when the class is instantiated.
    It sets up the initial state of the object.


    Args:
        self: Represent the instance of the class

    Returns:
        The __createframe function

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = None
        self.ListedOrModified = None
        self.dateStart = None
        self.dateEnd = None
        self.select = None
        self.file_name = None
        self.append_file = None

        self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")

    def __ShowGui(self, layout, text):

        """
    The __ShowGui function is a helper function that creates the GUI window and displays it to the user.
    It takes in two parameters: layout, which is a list of lists containing all the elements for each row;
    and text, which is a string containing what will be displayed as the title of the window. The
__ShowGui
```

```
    method then uses these parameters to create an instance of sg.Window with all its attributes set accordingly.

    Args:
        self: Refer to the current class instance
        layout: Pass the layout of the window to be created
        text: Set the title of the window

    Returns:
        A dictionary of values

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
                           finalize=True,
                           icon=ImageLoader("taskbar_icon.ico"))

        while True:
            event, values = window.read()

            if event == "Submit":
                try:
                    self.__SetValues(values)
                    break
                except Exception as e:
                    print(e)
                    RESTError(993)
                    raise SystemExit(993)
            elif event == sg.WIN_CLOSED or event == "Quit":
                break

        window.close()

    @staticmethod
    def __CreateFrame():
        """
    The __CreateFrame function creates the GUI layout for the application.
        The function returns a list of lists that contains all the elements to be displayed in the window.
        Each element is defined by its type and any additional parameters needed to define it.

    Args:

    Returns:
        A list of lists, which is used to create the gui

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        sg.theme('Default1')

        line00 = [sg.HSeparator()]

        line0 = [sg.Image(ImageLoader("logo.png")),
                 sg.Push(),
                 sg.Text("Utah Real Estate Utility", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push(),
                 sg.Push()]

        line1 = [sg.HSeparator()]

        line2 = [sg.Text("MLS Status : ", size=(15, None), justification="Right"),
                 sg.DropDown(default_value="Active", values=["Active", "Closed"], key="-status-", size=(31, 1))]

        line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
                 sg.DropDown(default_value="Listing Date", values=["Listing Date", "Modification Date", "Close Date"],
                             key="-type-", size=(31, 1))]

        line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"), key="-DateStart-",
                          disabled=False, size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-DateStart-")]

        line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
                 sg.Input(default_text=(date.today().strftime("%Y-%m-%d")), key="-DateEnd-", disabled=False,
                          size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-end_date-',
target="-DateEnd-")]

        line7 = [sg.HSeparator()]

        line8 = [sg.Push(),
                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
                 sg.Push()]

        line9 = [sg.HSeparator()]

        line10 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
                  sg.Input(default_text="", key="-AppendingFile-", disabled=True,
                           size=(20, 1)),
                  sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
                                target="-AppendingFile-")]

        line11 = [sg.HSeparator()]

        line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]

        layout = [line00, line0, line1, line2, line3, line4, line5, line7, line8, line9, line10, line11,
                  line12]

        return layout

    def __SetValues(self, values):

        """
    The __SetValues function is used to set the values of the variables that are used in the
       __GetData function. The values are passed from a dictionary called 'values' which is created
       by parsing through an XML file using ElementTree. This function also sets default values for
       some of these variables if they were not specified in the XML file.

    Args:
        self: Represent the instance of the class
        values: Pass the values from the gui to this function

    Returns:
        A dictionary with the following keys:

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.StandardStatus = values["-status-"]

        self.ListedOrModified = values["-type-"]

        if values["-DateStart-"] != "":
            self.dateStart = values["-DateStart-"]
        else:
            self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")

        if values["-DateEnd-"] != "":
            self.dateEnd = values["-DateEnd-"]
        else:
            self.dateEnd = (date.today()).strftime("%Y-%m-%d")

        self.select = None

        if values["-append_file-"] != "":
            self.append_file = str(values["-append_file-"])
        else:
            self.append_file = None


class UtahRealEstateMain:

    def __init__(self, siteClass):

        """
    The __init__ function is the first function that runs when an object of this class is created.
    It sets up all the variables and functions needed for this class to work properly.

    Args:
        self: Represent the instance of the class
        siteClass: Determine which site to pull data from

    Returns:
        Nothing
```

```

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        self.dataframe = None
        self.__batches = 0
        self.__siteClass = siteClass
        self.__headerDict = None
        self.__parameterString = ""
        self.__appendFile = None
        self.__dateStart = None
        self.__dateEnd = None
        self.__restDomain = settings.settingURERestDomain
        self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
            "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
        self.filePath = Path(os.path.expanduser('~/Documents')).joinpath("GardnerUtilData").joinpath(
            "Security").joinpath("auth.json")
        self.key = None
        self.__record_val = None

        try:
            self.mainFunc()
        except KeyError as e:
            # This allows for user cancellation of the program using the quit button
            if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
                RESTError(1101)
                print(e)
                pass
            else:
                pass
        except Exception as e:
            print(e)
            RESTError(1001)
            raise SystemExit(1001)

    def mainFunc(self):

        """
    The mainFunc function is the main function of this module. It will be called by the GUI when a user
clicks on
    the &quot;Run&quot; button in the GUI. The mainFunc function should contain all of your code for
running your program, and it
    should return a dataframe that contains all the data you want to display in your final report.

    Args:
        self: Reference the object itself

    Returns:
        A dataframe

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        passFlag = False

        while not passFlag:
            if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
                try:
                    f = open(self.keyPath, "rb")
                    key = f.readline()
                    f.close()
                    f = open(self.filePath, "rb")
                    authDict = json.load(f)
                    fernet = Fernet(key)
                    authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
                    self.__headerDict = {authDict["ure"]["parameter"]: authkey}
                    passFlag = True
                except Exception as e:
                    print(
                        f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
                    AuthUtil()
            else:
                AuthUtil()

        self.__ParameterCreator()

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Param String =
{self.__parameterString}")
        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain = {self.__restDomain}")

        self.__getCountUI()

        if self.__record_val is None:
            self.__record_val = 0

        self.__batches = BatchCalculator(self.__record_val, None)

        print(
            f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches = {self.__batches} | Rows {self.__record_val}")

        if self.__batches != 0:
            startTime = datetime.datetime.now().replace(microsecond=0)
            eventReturn = BatchInputGui(self.__batches, self.__record_val)
            if eventReturn == "Continue":
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches sent to server")
                BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
                                                  ParameterDict=self.__parameterString,
                                                  HeaderDict=self.__headerDict,
                                                  BatchesNum=self.__batches,
                                                  Type="utah_real_estate")
                BatchGuiObject.BatchGuiShow()
                self.dataframe = BatchGuiObject.dataframe
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Dataframe retrieved with {self.dataframe.shape[0]} rows and {self.dataframe.shape[1]} columns in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) - startTime).total_seconds()))}")
                FileSaver("ure", self.dataframe, self.__appendFile)
            else:
                print(
                    f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for {self.__batches} batches canceled by user")
        else:
            RESTError(994)
            raise SystemExit(994)

    def __ParameterCreator(self):
        """
    The __ParameterCreator function is used to create the filter string for the ReST API call.
    The function takes in a siteClass object and extracts all of its parameters into a dictionary.
    It then creates an appropriate filter string based on those parameters.

    Args:
        self: Bind the object to the class

    Returns:
        A string to be used as the parameter in the api call

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        filter_string = ""

        __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
                         not key.startswith('__') and not callable(key)}

        self.__appendFile = __Source_dict["append_file"]
        __Source_dict.pop("append_file")

        temp_dict = copy.copy(__Source_dict)
        for key, value in temp_dict.items():
            if value is None:
                __Source_dict.pop(key)
            else:
                pass

        if __Source_dict["ListedOrModified"] == "Listing Date":
            filter_string = f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContractDate%20le%20{__Source_dict['dateEn
        elif __Source_dict["ListedOrModified"] == "Modification Date":
            filter_string = f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20and%20ModificationTimestamp%20le%20{__Sour
        elif __Source_dict["ListedOrModified"] == "Close Date":
            filter_string = f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{__Source_dict['dateEnd']}"

        filter_string = filter_string +
f"%20and%20StandardStatus%20has%20Odata.Models.StandardStatus'{__Source_dict['StandardStatus']}'"

        self.__parameterString = filter_string

    def __getCount(self):
        """
    The __getCount function is used to determine the number of records that will be returned by the query.
    This function is called when a user calls the count() method on a ReST object. The __getCount function
uses
    the $count parameter in OData to return only an integer value representing how many records would be
returned
    by the query.

    Args:
        self: Represent the instance of the class

    Returns:
        The number of records in the data set

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        __count_resp = None

        try:
            __count_resp = requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
                                        headers=self.__headerDict)

        except requests.exceptions.Timeout as e:
            print(e)
            RESTError(790)
            raise SystemExit(790)
        except requests.exceptions.TooManyRedirects as e:
            print(e)
            RESTError(791)
            raise SystemExit(791)
        except requests.exceptions.MissingSchema as e:
            print(e)
            RESTError(1101)
        except requests.exceptions.RequestException as e:
            print(e)
            RESTError(405)
            raise SystemExit(405)

        self.__record_val = int(__count_resp.json()["@odata.count"])

    def __getCountUI(self):

        """
    The __getCountUI function is a wrapper for the __getCount function.
    It creates a progress window and updates it while the __getCount function runs.
    The purpose of this is to keep the GUI responsive while running long processes.

    Args:
        self: Represent the instance of the class

    Returns:
        A popupwrapped object

    Doc Author:
        Willem van der Schans, Trelent AI
    """
        uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)

        threadGui = threading.Thread(target=self.__getCount,
                                     daemon=False)
        threadGui.start()

        while threadGui.is_alive():
            uiObj.textUpdate()
            uiObj.windowPush()
        else:
            uiObj.stopWindow()
```