

Gardner API Utility Documentation

1.2.0

Willem van der Schans, DoxyGen

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 AuthUtil.AuthUtil Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 __init__()	6
3.1.3 Member Function Documentation	8
3.1.3.1 __CreateFrame()	8
3.1.3.2 __SetValues()	10
3.1.3.3 __ShowGui()	12
3.1.4 Member Data Documentation	14
3.1.4.1 append_file	14
3.1.4.2 file_name	14
3.1.4.3 filePath	15
3.1.4.4 jsonDict	15
3.1.4.5 k	15
3.1.4.6 keyFlag	15
3.1.4.7 keyPath	15
3.1.4.8 ListedOrModified	15
3.1.4.9 outcomeText	16
3.1.4.10 passFlagCm	16
3.1.4.11 passFlagUre	16
3.1.4.12 popupFlag	16
3.1.4.13 StandardStatus	16
3.2 BatchProcessing.BatchProcessorConstructionMonitor Class Reference	17
3.2.1 Detailed Description	17
3.2.2 Constructor & Destructor Documentation	17
3.2.2.1 __init__()	18
3.2.3 Member Function Documentation	19
3.2.3.1 ConstructionMonitorProcessor()	19
3.2.3.2 FuncSelector()	20
3.2.4 Member Data Documentation	21
3.2.4.1 __columnSelection	21
3.2.4.2 __dateTracker	21
3.2.4.3 __headerDict	22

3.2.4.4	<code>__maxRequests</code>	22
3.2.4.5	<code>__numBatches</code>	22
3.2.4.6	<code>__parameterDict</code>	22
3.2.4.7	<code>__requestCalls</code>	22
3.2.4.8	<code>__requestCount</code>	22
3.2.4.9	<code>__restDomain</code>	23
3.2.4.10	<code>dataframe</code>	23
3.2.4.11	<code>valueObject</code>	23
3.3	BatchProcessing.BatchProcessorUtahRealEstate Class Reference	23
3.3.1	Detailed Description	24
3.3.2	Constructor & Destructor Documentation	24
3.3.2.1	<code>__init__()</code>	24
3.3.3	Member Function Documentation	25
3.3.3.1	<code>BatchProcessingUtahRealestateCom()</code>	25
3.3.3.2	<code>FuncSelector()</code>	26
3.3.4	Member Data Documentation	27
3.3.4.1	<code>__headerDict</code>	27
3.3.4.2	<code>__numBatches</code>	27
3.3.4.3	<code>__parameterString</code>	27
3.3.4.4	<code>__restDomain</code>	27
3.3.4.5	<code>dataframe</code>	28
3.3.4.6	<code>valueObject</code>	28
3.4	BatchProgressGUI.BatchProgressGUI Class Reference	28
3.4.1	Detailed Description	29
3.4.2	Constructor & Destructor Documentation	29
3.4.2.1	<code>__init__()</code>	29
3.4.3	Member Function Documentation	30
3.4.3.1	<code>BatchGuiShow()</code>	30
3.4.3.2	<code>createGui()</code>	31
3.4.3.3	<code>CreateProgressLayout()</code>	33
3.4.3.4	<code>ProgressUpdater()</code>	34
3.4.3.5	<code>TimeUpdater()</code>	35
3.4.3.6	<code>ValueChecker()</code>	36
3.4.4	Member Data Documentation	37
3.4.4.1	<code>__batch_counter</code>	37
3.4.4.2	<code>__batches</code>	38
3.4.4.3	<code>__columnSelection</code>	38
3.4.4.4	<code>__headerDict</code>	38
3.4.4.5	<code>__layout</code>	38

3.4.4.6	<code>__parameterDict</code>	38
3.4.4.7	<code>__restDomain</code>	38
3.4.4.8	<code>__type</code>	39
3.4.4.9	<code>__window</code>	39
3.4.4.10	<code>dataframe</code>	39
3.5	Core.CFBP Class Reference	39
3.5.1	Detailed Description	40
3.5.2	Constructor & Destructor Documentation	40
3.5.2.1	<code>__init__()</code>	40
3.5.3	Member Function Documentation	41
3.5.3.1	<code>__dataGetter()</code>	41
3.5.3.2	<code>__showUi()</code>	42
3.5.4	Member Data Documentation	43
3.5.4.1	<code>link</code>	44
3.5.4.2	<code>state_arg</code>	44
3.5.4.3	<code>uiString</code>	44
3.5.4.4	<code>year_arg</code>	44
3.6	Core.ConstructionMonitorInit Class Reference	44
3.6.1	Detailed Description	45
3.6.2	Constructor & Destructor Documentation	45
3.6.2.1	<code>__init__()</code>	45
3.6.3	Member Function Documentation	47
3.6.3.1	<code>__CreateFrame()</code>	47
3.6.3.2	<code>__SetValues()</code>	49
3.6.3.3	<code>__ShowGui()</code>	50
3.6.4	Member Data Documentation	52
3.6.4.1	<code>append_file</code>	52
3.6.4.2	<code>auth_key</code>	53
3.6.4.3	<code>dateEnd</code>	53
3.6.4.4	<code>dateStart</code>	53
3.6.4.5	<code>rest_domain</code>	53
3.6.4.6	<code>size</code>	53
3.6.4.7	<code>SourceInclude</code>	53
3.6.4.8	<code>ui_flag</code>	54
3.7	Core.ConstructionMonitorMain Class Reference	54
3.7.1	Detailed Description	54
3.7.2	Constructor & Destructor Documentation	55
3.7.2.1	<code>__init__()</code>	55
3.7.3	Member Function Documentation	56

3.7.3.1 __getCount()	56
3.7.3.2 __getCountUI()	57
3.7.3.3 __ParameterCreator()	59
3.7.3.4 mainFunc()	60
3.7.4 Member Data Documentation	62
3.7.4.1 __appendFile	62
3.7.4.2 __batches	63
3.7.4.3 __columnSelection	63
3.7.4.4 __headerDict	63
3.7.4.5 __parameterDict	63
3.7.4.6 __record_val	63
3.7.4.7 __restDomain	63
3.7.4.8 __search_id	64
3.7.4.9 __siteClass	64
3.7.4.10 __ui_flag	64
3.7.4.11 dataframe	64
3.8 DataTransfer.DataTransfer Class Reference	64
3.8.1 Detailed Description	65
3.8.2 Constructor & Destructor Documentation	65
3.8.2.1 __init__()	65
3.8.3 Member Function Documentation	65
3.8.3.1 getValue()	66
3.8.3.2 setValue()	66
3.8.3.3 whileValue()	67
3.8.4 Member Data Documentation	68
3.8.4.1 __value	68
3.9 FileSaver.FileSaver Class Reference	68
3.9.1 Detailed Description	69
3.9.2 Constructor & Destructor Documentation	69
3.9.2.1 __init__()	69
3.9.3 Member Function Documentation	70
3.9.3.1 getPath()	71
3.9.4 Member Data Documentation	71
3.9.4.1 appendFlag	71
3.9.4.2 data	71
3.9.4.3 dataAppending	72
3.9.4.4 docPath	72
3.9.4.5 fileName	72
3.9.4.6 outputFrame	72

3.9.4.7 primaryKey	72
3.9.4.8 uiFlag	72
3.10 API_Calls.Initializer.initializer Class Reference	73
3.10.1 Detailed Description	73
3.10.2 Constructor & Destructor Documentation	73
3.10.2.1 __init__()	73
3.10.3 Member Function Documentation	74
3.10.3.1 __CreateFrame()	75
3.10.3.2 __ShowGui()	76
3.10.4 Member Data Documentation	78
3.10.4.1 classObj	78
3.11 PopupWrapped.PopupWrapped Class Reference	79
3.11.1 Detailed Description	79
3.11.2 Constructor & Destructor Documentation	79
3.11.2.1 __init__()	80
3.11.3 Member Function Documentation	81
3.11.3.1 __createLayout()	81
3.11.3.2 __createWindow()	82
3.11.3.3 openFile()	85
3.11.3.4 stopWindow()	85
3.11.3.5 textUpdate()	86
3.11.3.6 windowPush()	87
3.11.4 Member Data Documentation	87
3.11.4.1 __counter	87
3.11.4.2 __docpath	87
3.11.4.3 __error	88
3.11.4.4 __errorFlag	88
3.11.4.5 __layout	88
3.11.4.6 __text	88
3.11.4.7 __thread	88
3.11.4.8 __type	88
3.11.4.9 __windowObj	89
3.12 Core.realtorCom Class Reference	89
3.12.1 Detailed Description	89
3.12.2 Constructor & Destructor Documentation	90
3.12.2.1 __init__()	90
3.12.3 Member Function Documentation	91
3.12.3.1 __dataUpdater()	91
3.12.3.2 __linkGetter()	92

3.12.3.3 __showUi()	93
3.12.4 Member Data Documentation	94
3.12.4.1 __idDict	94
3.12.4.2 __last_date	95
3.12.4.3 __linkDict	95
3.12.4.4 __page_html	95
3.12.4.5 __update_date	95
3.12.4.6 dfCounty	95
3.12.4.7 dfState	95
3.12.4.8 dfZip	96
3.12.4.9 uiString	96
3.13 Settings.settings Class Reference	96
3.13.1 Detailed Description	96
3.13.2 Member Data Documentation	96
3.13.2.1 settingCFBPLink	97
3.13.2.2 settingCMRestDomain	97
3.13.2.3 settingDownloadSourceLink	97
3.13.2.4 settingGenerationToolLink	97
3.13.2.5 settingGithubApiUrl	97
3.13.2.6 settingRealtorLink	98
3.13.2.7 settingURERestDomain	98
3.13.2.8 settingVersion	98
3.14 Core.UtahRealEstateInit Class Reference	98
3.14.1 Detailed Description	99
3.14.2 Constructor & Destructor Documentation	99
3.14.2.1 __init__()	99
3.14.3 Member Function Documentation	100
3.14.3.1 __CreateFrame()	100
3.14.3.2 __SetValues()	102
3.14.3.3 __ShowGui()	103
3.14.4 Member Data Documentation	105
3.14.4.1 append_file	105
3.14.4.2 dateEnd	106
3.14.4.3 dateStart	106
3.14.4.4 file_name	106
3.14.4.5 ListedOrModified	106
3.14.4.6 select	106
3.14.4.7 StandardStatus	106
3.15 Core.UtahRealEstateMain Class Reference	107

3.15.1 Detailed Description	107
3.15.2 Constructor & Destructor Documentation	107
3.15.2.1 __init__()	108
3.15.3 Member Function Documentation	109
3.15.3.1 __getCount()	109
3.15.3.2 __getCountUI()	110
3.15.3.3 __ParameterCreator()	111
3.15.3.4 mainFunc()	113
3.15.4 Member Data Documentation	115
3.15.4.1 __appendFile	115
3.15.4.2 __batches	115
3.15.4.3 __dateEnd	115
3.15.4.4 __dateStart	116
3.15.4.5 __headerDict	116
3.15.4.6 __parameterString	116
3.15.4.7 __record_val	116
3.15.4.8 __restDomain	116
3.15.4.9 __siteClass	116
3.15.4.10 dataframe	117
3.15.4.11 filePath	117
3.15.4.12 key	117
3.15.4.13 keyPath	117
4 File Documentation	119
4.1 __init__.py	119
4.2 _main_.py	119
4.3 AuthUtil.py	119
4.4 BatchProcessing.py	123
4.5 DataSupportFunctions.py	127
4.6 FileSaver.py	127
4.7 Settings.py	129
4.8 versionChecker.py	129
4.9 ErrorPopup.py	130
4.10 ErrorPrint.py	130
4.11 Logger.py	130
4.12 RESTError.py	131
4.13 BatchGui.py	133
4.14 BatchProgressGUI.py	135
4.15 DataTransfer.py	139

4.16 ImageLoader.py	140
4.17 PopupWrapped.py	140
4.18 Initializer.py	144
4.19 CFBP/Core.py	147
4.20 ConstructionMonitor/Core.py	149
4.21 Realtor/Core.py	155
4.22 UtahRealEstate/Core.py	157

Chapter 1

Class Index

1.1 Class List

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

AuthUtil.AuthUtil	5
BatchProcessing.BatchProcessorConstructionMonitor	17
BatchProcessing.BatchProcessorUtahRealEstate	23
BatchProgressGUI.BatchProgressGUI	28
Core.CFBP	39
Core.ConstructionMonitorInit	44
Core.ConstructionMonitorMain	54
DataTransfer.DataTransfer	64
FileSaver.FileSaver	68
API_Calls.Initializer.initializer	73
PopupWrapped.PopupWrapped	79
Core.realtorCom	89
Settings.settings	96
Core.UtahRealEstateInit	98
Core.UtahRealEstateMain	107

Chapter 2

File Index

2.1 File List

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

__init__.py	119
main.py	119
AuthUtil.py	119
BatchProcessing.py	123
DataSupportFunctions.py	127
FileSaver.py	127
Settings.py	129
versionChecker.py	129
ErrorPopup.py	130
ErrorPrint.py	130
Logger.py	130
RESTError.py	131
BatchGui.py	133
BatchProgressGUI.py	135
DataTransfer.py	139
ImageLoader.py	140
PopupWrapped.py	140
Initializer.py	144
CFBP/Core.py	147
ConstructionMonitor/Core.py	149
Realtor/Core.py	155
UtahRealEstate/Core.py	157

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 " \
        "Submitting will overwrite any old values in an unrecoverable manner."
00047
00048         if os.path.exists(self.filePath):
00049             pass
00050         else:
00051             if os.path.exists(Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData")):
00052                 os.mkdir(self.filePath)
00053             else:
00054                 os.mkdir(Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData"))
00055                 os.mkdir(self.filePath)
00056
00057         if os.path.exists(self.keyPath):
00058             pass
00059         else:
00060             if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00061                 os.mkdir(self.keyPath)
00062             else:
```

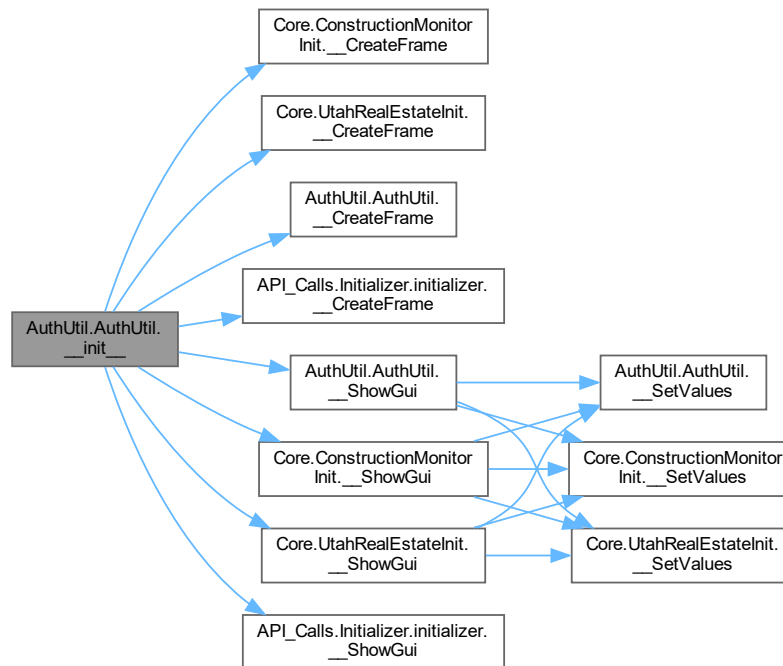


```

00064         os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
00065         os.mkdir(self.keyPath)
00066
00067     if os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
00068         try:
00069             f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00070             self.k = f.readline()
00071             f.close()
00072         except Exception as e:
00073             print(e)
00074             RESTError(402)
00075             raise SystemExit(402)
00076     else:
00077         self.k = Fernet.generate_key()
00078         f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
00079         f.write(self.k)
00080         f.close()
00081
00082     try:
00083         os.remove(self.filePath.joinpath("auth.json"))
00084     except Exception as e:
00085         # Logging
00086         print(
00087             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..."
00088         )
00089         pass
00090
00091     f = open(self.filePath.joinpath("auth.json"), "wb")
00092     f.close()
00093     self.keyFlag = False
00094
00095     self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")
00096
00097     try:
00098         ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), 2)
00099     except Exception as e:
00100         # Logging
00101         print(
00102             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..."
00103         )
00104         pass

```

Here is the call graph for this function:



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

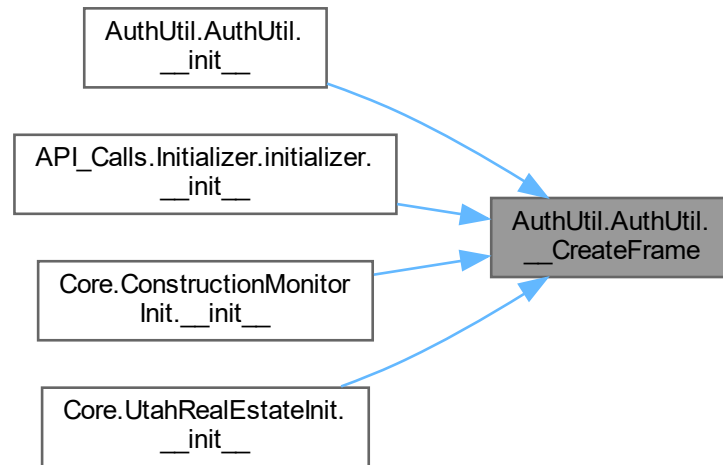
Definition at line 235 of file [AuthUtil.py](#).

```

00235     def __CreateFrame(self):
00236         """
00237         The __CreateFrame function creates the GUI layout for the Authentication Utility.
00238         It is called by __init__ and returns a list of lists that contains all the elements
00239         that will be displayed in the window.
00240
00241         Args:
00242             self: Access the class attributes and methods
00243
00244         Returns:
00245             A list of lists
00246
00247         Doc Author:
00248             Trelent
00249         """
00250         sg.theme('Default1')
00251
00252         line00 = [sg.HSeparator()]
00253
00254         line0 = [sg.Image(ImageLoader("logo.png")),
00255                 sg.Push(),
00256                 sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"),
justification="center"),
00257                 sg.Push(),
00258                 sg.Push()]
00259
00260         line1 = [sg.HSeparator()]
00261
00262         line2 = [sg.Push(),
00263                 sg.Text("Utah Real Estate API Key: ", justification="center"),
00264                 sg.Push()]
00265
00266         line3 = [sg.Push(),
00267                 sg.Input(default_text="123", key="-ureAuth-", disabled=False,
00268                        size=(40, 1)),
00269                 sg.Push()]
00270
00271         line4 = [sg.HSeparator()]
00272
00273         line5 = [sg.Push(),
00274                 sg.Text("Construction Monitor HTTP BASIC Key: ", justification="center"),
00275                 sg.Push()]
00276
00277         line6 = [sg.Push(),
00278                 sg.Input(default_text="Basic 123", key="-cmAuth-", disabled=False,
00279                        size=(40, 1)),
00280                 sg.Push()]
00281
00282         line7 = [sg.HSeparator()]
00283
00284         line8 = [sg.Push(),
00285                 sg.Text(self.outcomeText, justification="center"),
00286                 sg.Push()]
00287
00288         line9 = [sg.HSeparator()]
00289
00290         line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00291
00292         layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]
00293
00294         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 "OK" 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](#).

```
00104     def __SetValues(self, values):
00105
00106         """
00107         The __SetValues function is called when the user clicks on the "OK" button in the window.
00108         It takes a dictionary of values as an argument, and then uses those values to update
00109         the auth.json file with new keys for both Utah Real Estate and Construction Monitor.
00110
```

```

00111     Args:
00112         self: Make the function a method of the class
00113         values: Store the values that are entered into the form
00114
00115     Returns:
00116         A dictionary of the values entered by the user
00117
00118     Doc Author:
00119         Willem van der Schans, Trelent AI
00120     """
00121     ureCurrent = None
00122     cmCurrent = None
00123     keyFile = None
00124     self.popupFlag = False
00125
00126     fernet = Fernet(self.k)
00127
00128     try:
00129         f = open(self.filePath.joinpath("auth.json"), "r")
00130         keyFile = json.load(f)
00131         fileFlag = True
00132     except:
00133         fileFlag = False
00134
00135     # Try initial decoding, if fails pass and write new keys and files
00136     if fileFlag:
00137         try:
00138             ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
00139         except Exception as e:
00140             # Logging
00141             print(
00142                 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")
00143             ureCurrent = None
00144
00145         try:
00146             cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
00147         except Exception as e:
00148             # Logging
00149             print(
00150                 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")
00151             cmCurrent = None
00152
00153         if values["-ureAuth-"] != "":
00154             self.jsonDict.update(
00155                 {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
00156             self.passFlagUre = True
00157         elif ureCurrent is not None:
00158             self.jsonDict.update(
00159                 {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(ureCurrent.encode()).decode()}})
00160             self.passFlagUre = True
00161         else:
00162             pass
00163
00164         if values["-cmAuth-"] != "":
00165             if values["-cmAuth-"].startswith("Basic"):
00166                 self.jsonDict.update(
00167                     {"cm": {"parameter": "Authorization",
00168                             "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
00169                 self.passFlagCm = True
00170             else:
00171                 PopupWrapped("Please make sure you provide a HTTP Basic Auth key for construction
Monitor",
00172                             windowType="AuthError")
00173                 self.popupFlag = True
00174                 pass
00175         elif ureCurrent is not None:
00176             self.jsonDict.update(
00177                 {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(cmCurrent.encode()).decode()}})
00178             self.passFlagUre = True
00179         else:
00180             pass
00181
00182         if not self.passFlagUre and not self.passFlagCm:
00183             PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction

```

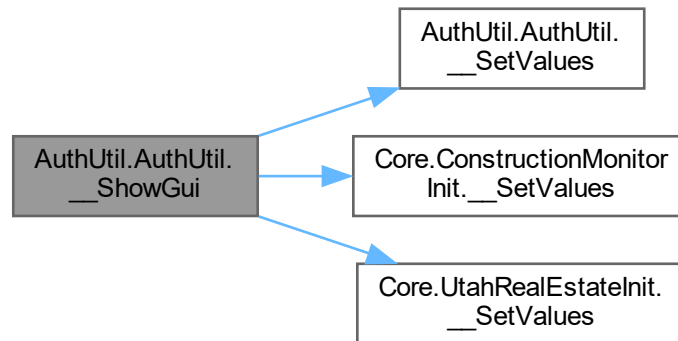

Definition at line 196 of file [AuthUtil.py](#).

```

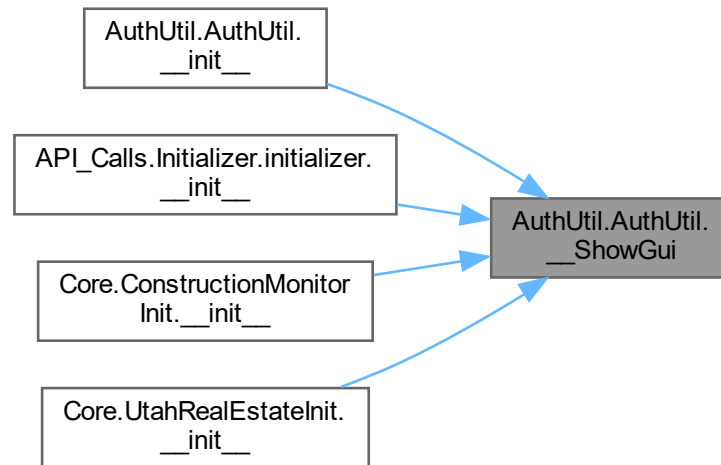
00196     def __ShowGui(self, layout, text):
00197
00198         """
00199         The __ShowGui function is a helper function that displays the GUI to the user.
00200         It takes in two arguments: layout and text. The layout argument is a list of lists,
00201         which contains all the elements that will be displayed on screen. The text argument
00202         is simply what will be displayed at the top of the window.
00203
00204         Args:
00205             self: Represent the instance of the class
00206             layout: Pass the layout of the gui to be displayed
00207             text: Set the title of the window
00208
00209         Returns:
00210             A window object
00211         """
00212         window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00213                             finalize=True,
00214                             icon=ImageLoader("taskbar_icon.ico"))
00215
00216         while not self.passFlagUre or not self.passFlagCm:
00217             event, values = window.read()
00218
00219             if event == "Submit":
00220                 try:
00221                     self.__SetValues(values)
00222                 except Exception as e:
00223                     print(e)
00224                     RESTError(993)
00225                 finally:
00226                     pass
00227             elif event == sg.WIN_CLOSED or event == "Quit":
00228                 break
00229             else:
00230                 pass
00231
00232         window.close()
00233
00234

```

Here is the call graph for this function:



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`.

```
00043 def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject):
00044
00045     """
00046     The __init__ function is the constructor for a class. It is called when an object of that class
00047     is created, and it sets up the attributes of that object. In this case, we are setting up our
00048     object to have a dataframe attribute (which will be used to store all of our data), as well as
00049     attributes for each parameter in our ReST call.
00050
00051     Args:
00052         self: Represent the instance of the class
00053         RestDomain: Specify the domain of the rest api
00054         NumBatches: Determine how many batches of data to retrieve
00055         ParameterDict: Pass in the parameters that will be used to make the api call
00056         HeaderDict: Pass the header dictionary from the main function to this class
00057         ColumnSelection: Determine which columns to pull from the api
00058         valueObject: Pass in the value object that is used to determine what values are returned
00059
00060     Returns:
00061         An object of the class
00062
00063     Doc Author:
00064         Willem van der Schans, Trelent AI
00065     """
00066     self.dataframe = None
00067     self.__numBatches = NumBatches
00068     self.__parameterDict = ParameterDict
00069     self.__restDomain = RestDomain
00070     self.__headerDict = HeaderDict
00071     self.__columnSelection = ColumnSelection
00072     self.valueObject = valueObject
00073     self.__maxRequests = 10000
00074     self.__requestCount = math.ceil(self.__numBatches / (self.__maxRequests /
int(self.__parameterDict['size'])))
00075     self.__requestCalls = math.ceil(self.__maxRequests / int(self.__parameterDict['size']))
00076     self.__dateTracker = None
00077
```

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](#).

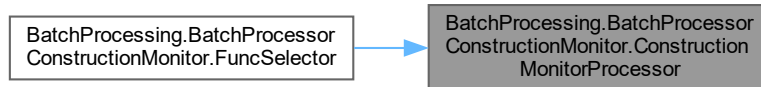
```
00094     def ConstructionMonitorProcessor(self, valueObject):
00095         """
00096         The ConstructionMonitorProcessor function will use requests to get data from
00097         ConstructionMontior.com's ReST API and store it into a pandas DataFrame object called __df (which
00098         is local). This
00099         process will be repeated until all the data has been collected from ConstructionMonitor.com's ReST
00099         API, at which point __df will contain all
00100
00100         Args:
00101             self: Represent the instance of the object itself
00102             valueObject: Update the progress bar in the gui
00103
00104         Returns:
00105             A dataframe
00106
00107         Doc Author:
00108             Willem van der Schans, Trelent AI
00109         """
00110         __df = None
00111         for callNum in range(0, self.__requestCount):
00112             self.__parameterDict["from"] = 0
00113
00114             if self.__requestCount > 1 and callNum != self.__requestCount - 1:
00115                 __batchNum = self.__requestCalls
00116                 if __df is None:
00117                     self.__dateTracker = str(date.today())
00118                 else:
00119                     self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00120             elif self.__requestCount == 1:
00121                 __batchNum = self.__numBatches
00122                 self.__dateTracker = str(date.today())
00123             else:
00124                 __batchNum = self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size']))
00125         - (
00126             self.__requestCount - 1)
00126             self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00127
00128             self.__parameterDict['dateEnd'] = self.__dateTracker
00129
00130             for record in range(0, int(math.ceil(__batchNum))):
00131                 if record != 0:
00132                     self.__parameterDict["from"] = record * int(self.__parameterDict["size"])
```

```

00133         response = requests.post(url=self.__restDomain,
00134                                 headers=self.__headerDict,
00135                                 json=self.__parameterDict)
00136
00137     counter = 0
00138     try:
00139         response = response.json()['hits']['hits']
00140     except KeyError as e:
00141         # Logging
00142         print(
00143             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}")
00145         continue
00146
00147     valueObject.setValue(valueObject.getValue() + 1)
00148
00149     if record == 0 and callNum == 0:
00150         __df = pd.json_normalize(response[counter]["_source"])
00151         __df["id"] = response[counter]['_id']
00152         __df["county"] = response[counter]["_source"]['county']['county_name']
00153         counter += 1
00154
00155     for i in range(counter, len(response)):
00156         __tdf = pd.json_normalize(response[i]["_source"])
00157         __tdf["id"] = response[i]['_id']
00158         __tdf["county"] = response[i]["_source"]['county']['county_name']
00159         __df = pd.concat([__df, __tdf], ignore_index=True)
00160
00161     if self.__columnSelection is not None:
00162         __col_list = StringToList(self.__columnSelection)
00163         __col_list.append("id")
00164         __col_list.append("county")
00165     else:
00166         pass
00167
00168     self.dataframe = __df
00169     valueObject.setValue(-999)
00170
00171

```

Here is the caller graph for this function:



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 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

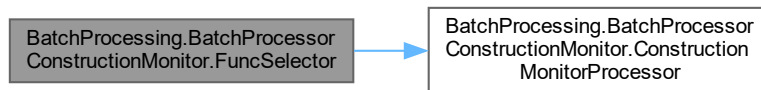
Doc Author:

Willem van der Schans, Trelent AI

Definition at line 78 of file [BatchProcessing.py](#).

```
00078     def FuncSelector(self):
00079         """
00080         The FuncSelector function is a function that takes the valueObject and passes it to the
00081         ConstructionMonitorProcessor function.
00082         The ConstructionMonitorProcessor function then uses this valueObject to determine which of its
00083         functions should be called.
00084
00085         Args:
00086             self: Represent the instance of the class
00087
00088         Returns:
00089             The result of the constructionmonitorprocessor function
00090
00091         Doc Author:
00092             Willem van der Schans, Trelent AI
00093         """
00094         self.ConstructionMonitorProcessor(self.valueObject)
```

Here is the call graph for this function:



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](#).

```
00174     def __init__(self, RestDomain, NumBatches, ParameterString, HeaderDict, valueObject):
00175         """
00176         The __init__ function is the constructor for a class. It is called when an object of that class
00177         is instantiated, and it sets up the attributes of that object. In this case, we are setting up
00178         the dataframe attribute to be None (which will be set later), and we are also setting up some
00179         other attributes which will help us make our API calls.
00180
00181         Args:
00182             self: Represent the instance of the class
00183             RestDomain: Specify the domain of the rest api
00184             NumBatches: Determine how many batches of data to pull from the api
00185             ParameterString: Pass the parameters to the rest api
00186             HeaderDict: Pass in the header information for the api call
00187             valueObject: Create a dataframe from the json response
00188
00189         Returns:
00190             The instance of the class
00191
00192         Doc Author:
00193             Willem van der Schans, Trelent AI
00194         """
00195         self.dataframe = None
00196         self.__numBatches = NumBatches
00197         self.__parameterString = ParameterString
00198         self.__restDomain = RestDomain
00199         self.__headerDict = HeaderDict
00200         self.valueObject = valueObject
00201
```

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](#).

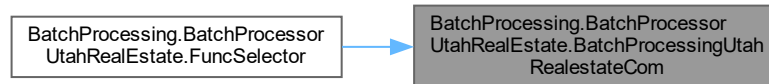
```
00219     def BatchProcessingUtahRealestateCom(self, valueObject):
00220         """
00221         The BatchProcessingUtahRealestateCom function is a function that takes in the valueObject and uses it
00222         to
00223         update the progress bar. It also takes in self, which contains all the necessary information for
00224         this
00225         function to work properly. The BatchProcessingUtahRealestateCom function will then use requests to
00226         get data from
00227         UtahRealestate.com's ReST API and store it into a pandas DataFrame object called __df (which is
00228         local). This
00229         process will be repeated until all the data has been collected from UtahRealestate.com's ReST API,
00230         at which point __df will contain all
00231
00232         Args:
00233         self: Represent the instance of the class
00234         valueObject: Pass the value of a progress bar to the function
00235
00236         Returns:
00237         A dataframe of the scraped data
00238
00239         Doc Author:
00240         Willem van der Schans, Trelent AI
00241         """
00242         __df = pd.DataFrame()
00243
00244         for batch in range(self.__numBatches):
00245
00246             if batch == 0:
00247                 response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200",
00248                                         headers=self.__headerDict)
00249
00250                 response_temp = response.json()
00251                 __df = pd.json_normalize(response_temp, record_path=['value'])
00252
00253             else:
00254                 response = requests.get(f"{self.__restDomain}{self.__parameterString}&top=200&$skip={batch
00255                                         * 200}",
00256                                         headers=self.__headerDict)
00257
00258                 response_temp = response.json()
```

```

00253         response_temp = pd.json_normalize(response_temp, record_path=['value'])
00254         __df = pd.concat([__df, response_temp], ignore_index=True)
00255
00256         valueObject.setValue(valueObject.getValue() + 1)
00257
00258         self.dataframe = __df
00259         valueObject.setValue(-999)

```

Here is the caller graph for this function:



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

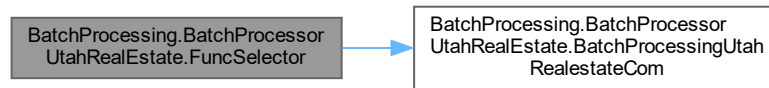
Definition at line 202 of file [BatchProcessing.py](#).

```

00202     def FuncSelector(self):
00203         """
00204         The FuncSelector function is a function that takes the valueObject as an argument and then calls the
00205         appropriate function based on what was selected in the dropdown menu. The valueObject is passed to each of
00206         these functions
00207         so that they can access all of its attributes.
00208
00209         Args:
00210             self: Represent the instance of the class
00211
00212         Returns:
00213             The function that is selected by the user
00214
00215         Doc Author:
00216             Willem van der Schans, Trelent AI
00217         """
00218         self.BatchProcessingUtahRealestateCom(self.valueObject)

```

Here is the call graph for this function:



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](#).

```
00019     def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None):
00020
00021         """
00022         The __init__ function is the first function that gets called when an object of this class is created.
00023         It initializes all the variables and sets up a layout for the GUI. It also creates a window to display
00024         the dataframe in.
00025
00026         Args:
00027             self: Represent the instance of the class
00028             BatchesNum: Determine the number of batches that will be created
00029             RestDomain: Specify the domain of the rest api
00030             ParameterDict: Pass the parameters of the request to the class
00031             HeaderDict: Store the headers of the dataframe
00032             Type: Determine the type of dataframe that is being created
00033             ColumnSelection: Select the columns to be displayed in the gui
00034
00035         Returns:
00036             Nothing
00037
00038         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

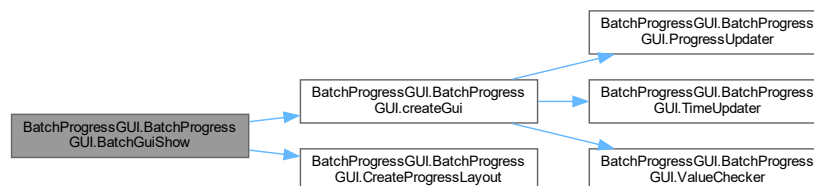
Definition at line 53 of file [BatchProgressGUI.py](#).

```

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

```

Here is the call graph for this function:



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 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

Definition at line 104 of file [BatchProgressGUI.py](#).

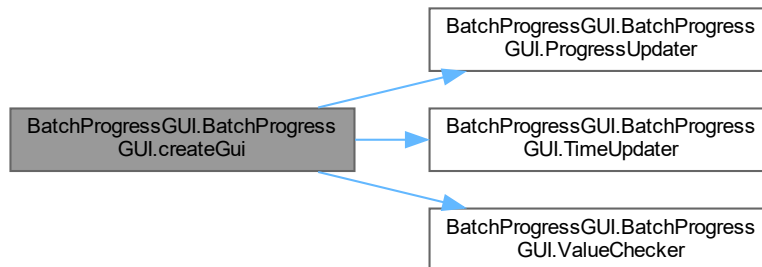
```
00104     def createGui(self, Sourcetype):
00105
00106         """
00107         The createGui function is the main function that creates the GUI.
00108         It takes in a type parameter which determines what kind of batch processor to use.
00109         The createGui function then sets up all 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
00113         these threads.
00114
00115         Args:
00116             self: Access the object itself
00117             Sourcetype: Determine which batch processor to use
00118
00119         Returns:
00120             The dataframe
00121
00122         Doc Author:
00123             Willem van der Schans, Trelent AI
00124         """
00125         self.__window = sg.Window('Progress', self.__layout, finalize=True,
00126                                   icon=ImageLoader("taskbar_icon.ico"))
00127
00128         start_time = datetime.datetime.now().replace(microsecond=0)
00129         update_text = f"Batch {0} completed"
00130         self.__window['--progress_text--'].update(update_text)
00131         self.__window['--progress_bar--'].update(0)
00132         self.__window['--time_est--'].update("Est time needed 00:00:00")
00133
00134         valueObj = DataTransfer()
00135         valueObj.setValue(0)
00136
00137         if Sourcetype == "construction_monitor":
00138             processorObject = BatchProcessorConstructionMonitor(RestDomain=self.__restDomain,
00139                                                                 NumBatches=self.__batches,
00140                                                                 ParameterDict=self.__parameterDict,
00141                                                                 HeaderDict=self.__headerDict,
00142                                                                 ColumnSelection=self.__columnSelection,
00143                                                                 valueObject=valueObj)
00144
00145         elif Sourcetype == "utah_real_estate":
00146             processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
```

```

00145                                     NumBatches=self.__batches,
00146                                     ParameterString=self.__parameterDict,
00147                                     HeaderDict=self.__headerDict,
00148                                     valueObject=valueObj)
00149
00150         threading.Thread(target=self.TimeUpdater,
00151                           args=(start_time,),
00152                           daemon=True).start()
00153         print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | TimeUpdater Thread
Successfully Started")
00154
00155         batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
00156                                             daemon=False)
00157         batchFuncThread.start()
00158         print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | BatchFunc Thread
Successfully Started")
00159         threading.Thread(target=self.ValueChecker,
00160                           args=(valueObj,),
00161                           daemon=False).start()
00162         print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | ValueChecker Thread
Successfully Started")
00163
00164         while True:
00165             self.ProgressUpdater(valueObj)
00166
00167             if valueObj.getValue() == -999:
00168                 break
00169
00170             window, event, values = sg.read_all_windows()
00171             if event.startswith('update'):
00172                 __key_to_update = event[len('update'):]
00173                 window[__key_to_update].update(values[event])
00174                 window.refresh()
00175                 pass
00176
00177             if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
00178                 break
00179
00180             time.sleep(0.1)
00181
00182         self.dataframe = processorObject.dataframe
00183         self.__window.close()
00184
00185         PopupWrapped(text="Api Request Completed", windowType="notice")
00186
00187

```

Here is the call graph for this function:



Here is the caller graph for this function:



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 = [
```

```

00096         sg.ProgressBar(max_value=self.__batches, bar_color=("#920303", "#C9c8c8"), orientation='h',
00097         size=(30, 20),
00098         key='--progress_bar--')]
00099
00100         layout = [__Line1, __Line2, __Line3]
00101
00102         self.__layout = layout
00103

```

Here is the caller graph for this function:



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](#).

```

00188     def ProgressUpdater(self, valueObj):
00189         """
00190         The ProgressUpdater function is a callback function that updates the progress bar and text
00191         in the GUI. It takes in one argument, which is an object containing information about the
00192         current batch number. The ProgressUpdater function then checks if this value has changed from
00193         the last time it was called (i.e., if we are on a new batch). If so, it updates both the progress
00194         bar and text with this new information.
00195
00196         Args:
00197             self: Make the progressupdater function an instance method
00198             valueObj: Get the current value of the batch counter
00199
00200         Returns:
00201             The value of the batch counter
00202

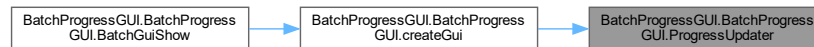
```

```

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:



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 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

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
00220         complete the current batch. It does this by reading the start_time variable passed in, getting the current
00221         time, calculating how much time has passed since start_time was set and then updating a timer string with
00222         that value. It then calculates an estimation of how long it will take to finish all batches based on how many
00223         batches have been completed so far.
00224
00225         Args:
00226             self: Make the function a method of the class
00227             start_time: Get the time when the function is called

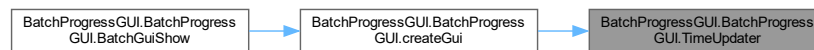
```

```

00227
00228     Returns:
00229         A string that is updated every 0
00230
00231     Doc Author:
00232         Willem van der Schans, Trelent AI
00233     """
00234     while True:
00235         if self.__batch_counter < self.__batches:
00236             __current_time = datetime.datetime.now().replace(microsecond=0)
00237             __passed_time = __current_time - start_time
00238             __timer_string = f"Time Elapsed {__passed_time}"
00239
00240             try:
00241                 self.__window.write_event_value('update--timer--', __timer_string)
00242             except AttributeError as e:
00243                 print(
00244                     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")
00245                 break
00246             __passed_time = __passed_time.total_seconds()
00247
00248             try:
00249                 __time_est = datetime.timedelta(
00250                     seconds=(__passed_time * (self.__batches / self.__batch_counter) -
__passed_time)).seconds
00251             except:
00252                 __time_est = datetime.timedelta(
00253                     seconds=(__passed_time * self.__batches - __passed_time)).seconds
00254
00255                 __time_est = time.strftime('%H:%M:%S', time.gmtime(__time_est))
00256
00257                 __end_string = f"Est time needed {__time_est}"
00258                 self.__window.write_event_value('update--time_est--', __end_string)
00259             else:
00260                 __end_string = f"Est time needed 00:00:00"
00261                 self.__window.write_event_value('update--time_est--', __end_string)
00262                 time.sleep(0.25)
00263
00264
00265
00266
00267

```

Here is the caller graph for this function:



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

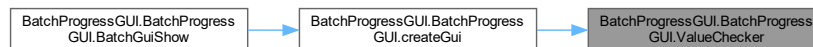
Doc Author:

Willem van der Schans, Trelent AI

Definition at line 268 of file [BatchProgressGUI.py](#).

```
00268     def ValueChecker(self, ObjectVal):
00269         """
00270         The ValueChecker function is a thread that checks the value of an object.
00271         It will check if the value has changed, and if it has, it will return True.
00272         If not, then it returns False.
00273
00274         Args:
00275             self: Represent the instance of the class
00276             ObjectVal: Get the value of the object
00277
00278         Returns:
00279             True if the value of the object has changed, and false if it hasn't
00280
00281         Doc Author:
00282             Willem van der Schans, Trelent AI
00283         """
00284         while True:
00285             time.sleep(0.3)
00286             if self.__batch_counter != ObjectVal.getValue():
00287                 self.__batch_counter = ObjectVal.getValue()
00288                 return True
00289             else:
00290                 return False
```

Here is the caller graph for this function:



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](#).

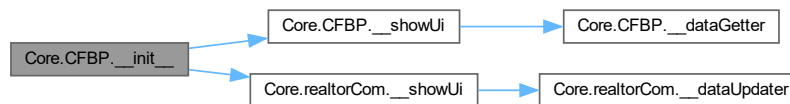
```
00017     def __init__(self, state_arg=None, year_arg=None):
00018         """
00019         The __init__ function is called when the class is instantiated.
00020         Its job is to initialize the object with some default values, and do any other setup that might be
00021         necessary.
00022         The __init__ function can take arguments, but it doesn't have to.
00023         Args:
00024             self: Represent the instance of the class
00025             state_arg: Set the state_arg attribute of the class
00026             year_arg: Set the year of data to be retrieved
00027         Returns:
00028             A popupwrapped object
00029         Doc Author:
00030             Willem van der Schans, Trelent AI
00031         """
00032         self.state_arg = state_arg
00033         self.year_arg = year_arg
00034         self.uiString = None
00035         self.link = None
00036
00037         eventReturn = confirmDialog()
00038         if eventReturn == "Continue":
00039             startTime = datetime.datetime.now().replace(microsecond=0)
00040             self.__showUi()
00041             print(
00042                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | API Link =
00043                 {self.link}")
00044             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.cfbp.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:

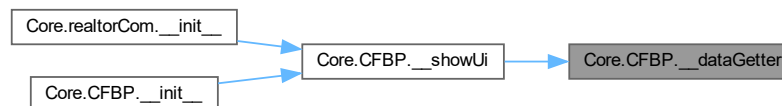
```

```

00095         A response object
00096
00097     Doc Author:
00098         Willem van der Schans, Trelent AI
00099     """
00100     arg_dict_bu = locals()
00101
00102     link = settings.settingCFBPLink
00103
00104     if self.state_arg is None:
00105         self.state_arg = "UT"
00106     else:
00107         pass
00108
00109     if self.year_arg is None:
00110         self.year_arg = str(datetime.date.today().year - 1)
00111     else:
00112         pass
00113
00114     passFlag = False
00115
00116     while not passFlag:
00117         self.link = link + f"states={self.state_arg}" + f"&years={self.year_arg}"
00118
00119         response = requests.get(self.link)
00120
00121         if response.status_code == 400:
00122             self.year_arg = int(self.year_arg) - 1
00123
00124         else:
00125             passFlag = True
00126
00127     RESTError(response)
00128     raise SystemExit(0)
00129

```

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

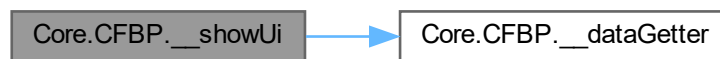
Definition at line 58 of file CFBP/Core.py.

```

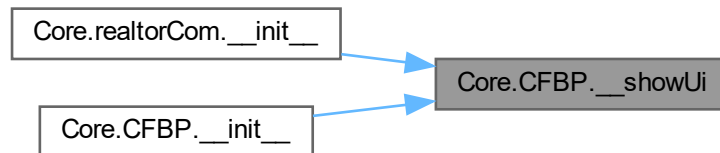
00058     def __showUi(self):
00059
00060         """
00061         The __showUi function is a function that creates a progress bar window.
00062         The __showUi function takes class variables and returns a windowobj.
00063
00064         Args:
00065             self: Represent the instance of the class
00066
00067         Returns:
00068             The uiobj variable
00069
00070         Doc Author:
00071             Willem van der Schans, Trelent AI
00072         """
00073         uiObj = PopupWrapped(text="Cenus Request running", windowType="progress", error=None)
00074
00075         threadGui = threading.Thread(target=self.__dataGetter,
00076                                     daemon=False)
00077         threadGui.start()
00078
00079         while threadGui.is_alive():
00080             uiObj.textUpdate()
00081             uiObj.windowPush()
00082         else:
00083             uiObj.stopWindow()
00084
00085

```

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

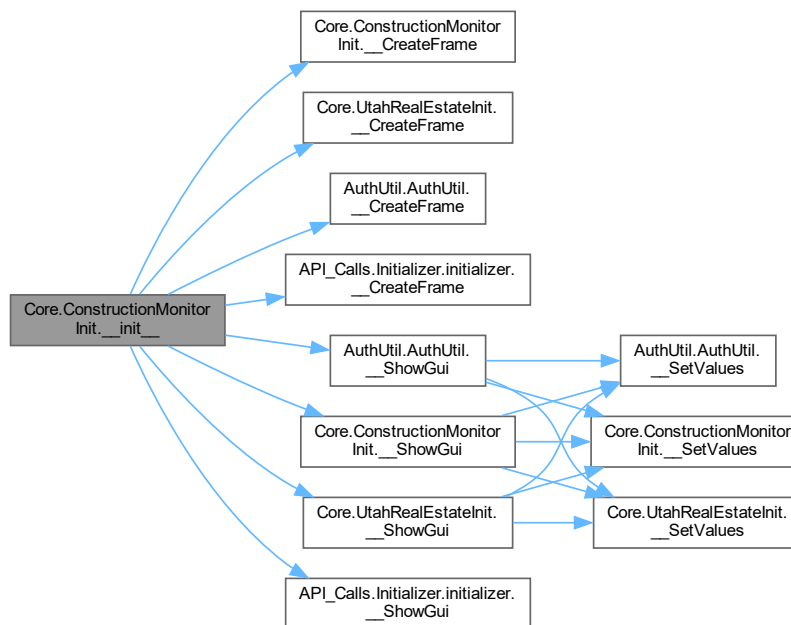
Definition at line 27 of file [ConstructionMonitor/Core.py](#).

```

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         Args:
00034             self: Represent the instance of the class
00035
00036         Returns:
00037             None
00038
00039         Doc Author:
00040             Willem van der Schans, Trelent AI
00041
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:



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](#).

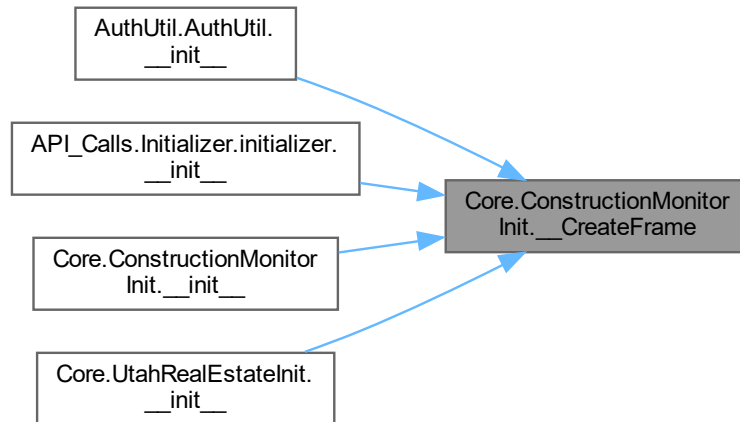
```
00117     def __CreateFrame():
00118
00119         """
00120         The __CreateFrame function creates the GUI layout for the application.
```

```

00121         The function returns a list of lists that contains all the elements to be displayed in the GUI
00122         window.
00123         This is done by creating each line as a list and then appending it to another list which will
00124         contain all lines.
00125
00126     Args:
00127         Returns:
00128             The layout for the gui
00129
00130     Doc Author:
00131         Willem van der Schans, Trelent AI
00132
00133     """
00134     sg.theme('Default1')
00135
00136     line00 = [sg.HSeparator()]
00137
00138     line0 = [sg.Image(ImageLoader("logo.png")),
00139              sg.Push(),
00140              sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"),
00141                    justification="center"),
00142              sg.Push(),
00143              sg.Push()]
00144
00145     line1 = [sg.HSeparator()]
00146
00147     line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00148             sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"),
00149                    key="-Cal-",
00150                    size=(20, 1)),
00151             sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]
00152
00153     line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00154             sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-",
00155                    size=(20, 1)),
00156             sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
00157                    target="-EndCal-")]
00158
00159     line5 = [sg.HSeparator()]
00160
00161     line6 = [sg.Push(),
00162             sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
00163             sg.Push()]
00164
00165     line7 = [sg.HSeparator()]
00166
00167     line8 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
00168             sg.Input(default_text="", key="-AppendingFile-", disabled=True,
00169                    size=(20, 1)),
00170             sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
00171                    target="-AppendingFile-")]
00172
00173     line9 = [sg.HSeparator()]
00174
00175     line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00176
00177     layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]
00178
00179     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 in 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

Definition at line 176 of file [ConstructionMonitor/Core.py](#).

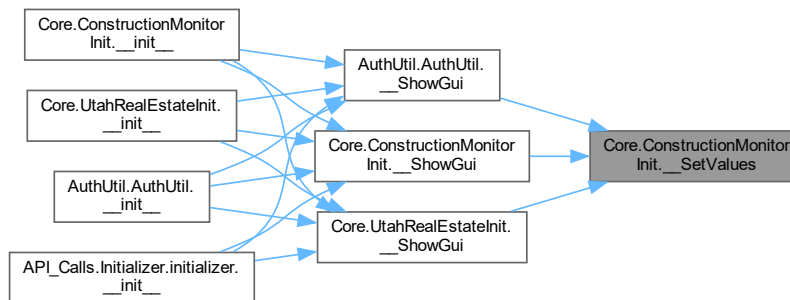
```
00176     def __SetValues(self, values):
00177
00178         """
00179         The __SetValues function is used to set the values of the variables that are used in the __GetData
00180         function.
00181         The __SetValues function takes a dictionary as an argument, and then sets each variable based on what
00182         is passed into
00183         the dictionary. The keys for this dictionary are defined by the user when they create their own
00184         instance of this class.
00185         """
```

```

00183     Args:
00184         self: Represent the instance of the class
00185         values: Pass in the values from the ui
00186
00187     Returns:
00188         A dictionary of values
00189
00190     Doc Author:
00191         Willem van der Schans, Trelent AI
00192     """
00193     self.size = 1000
00194
00195     if values["-Cal-"] != "":
00196         self.dateStart = values["-Cal-"]
00197     else:
00198         self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")
00199
00200     if values["-EndCal-"] != "":
00201         self.dateEnd = values["-EndCal-"]
00202     else:
00203         self.dateEnd = date.today().strftime("%Y-%m-%d")
00204
00205     self.rest_domain = settings.settingCMRestDomain
00206
00207     self.SourceInclude = None
00208
00209     if values["-append_file-"] != "":
00210         self.append_file = str(values["-append_file-"])
00211     else:
00212         self.append_file = None
00213
00214     self.ui_flag = True
00215
00216

```

Here is the caller graph for this function:



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

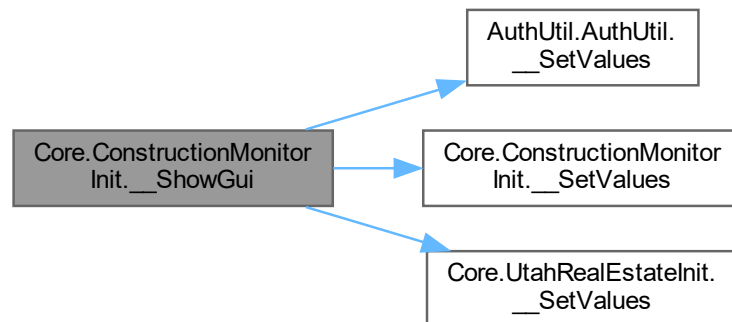
Doc Author:

Willem van der Schans, Trelent AI

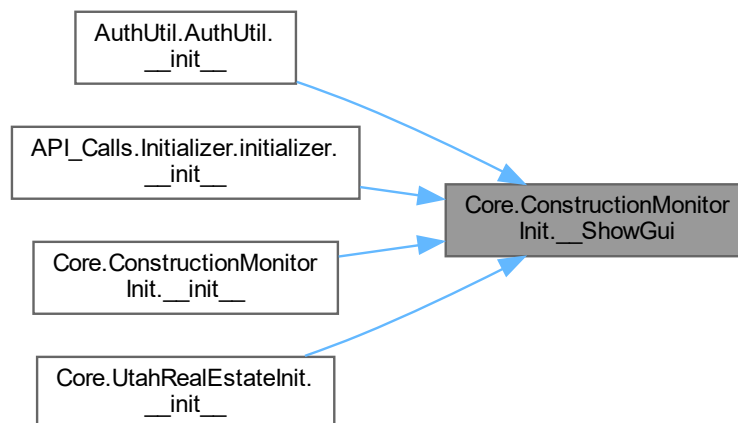
Definition at line 78 of file [ConstructionMonitor/Core.py](#).

```
00078     def __ShowGui(self, layout, text):
00079
00080         """
00081         The __ShowGui function is the main function that creates and displays the GUI.
00082         It takes in a layout, which is a list of lists containing all the elements to be displayed on screen.
00083         The text parameter specifies what title should appear at the top of the window.
00084
00085         Args:
00086             self: Refer to the current instance of a class
00087             layout: Determine what the gui will look like
00088             text: Set the title of the window
00089
00090         Returns:
00091             A dictionary of values
00092
00093         Doc Author:
00094             Willem van der Schans, Trelent AI
00095         """
00096         window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00097                           finalize=True,
00098                           icon=ImageLoader("taskbar_icon.ico"))
00099
00100         while True:
00101             event, values = window.read()
00102
00103             if event == "Submit":
00104                 try:
00105                     self.__SetValues(values)
00106                     break
00107                 except Exception as e:
00108                     print(e)
00109                     RESTError(993)
00110                     raise SystemExit(993)
00111             elif event == sg.WIN_CLOSED or event == "Quit":
00112                 break
00113
00114         window.close()
00115
```

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](#).

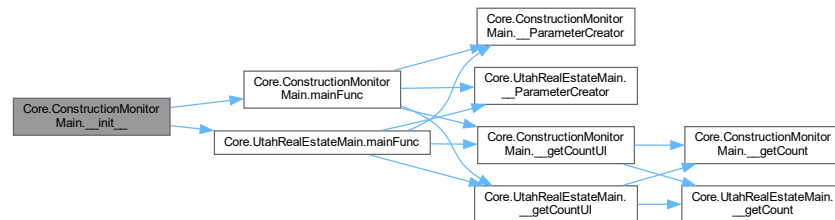
```
00219     def __init__(self, siteClass):
00220
00221         """
00222         The __init__ function is the first function that runs when an object of this class is created.
00223         It sets up all the variables and functions needed for this class to run properly.
00224
00225         Args:
00226             self: Represent the instance of the class
00227             siteClass: Identify the site that is being used
00228
00229         Returns:
00230             Nothing
00231
00232         Doc Author:
00233             Willem van der Schans, Trelent AI
00234         """
00235         self.__siteClass = siteClass
00236         self.__restDomain = None
00237         self.__headerDict = None
00238         self.__columnSelection = None
00239         self.__appendFile = None
00240
00241         self.__parameterDict = {}
00242         self.__search_id = None
00243         self.__record_val = 0
00244         self.__batches = 0
00245
00246         self.__ui_flag = None
00247
00248         self.dataframe = None
00249
00250         try:
00251             self.mainFunc()
00252         except SystemError as e:
00253             if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):
00254                 print(
00255                     f"ConstructionMonitor/Core.py | Error = {e} | Coerced SystemError in
00256                     ConstructionMonitorMain class")
00257                 pass
00258             except AttributeError as e:
00259                 # This allows for user cancellation of the program using the quit button
```

```

00260         if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
00261             RESTError(1101)
00262             print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Error {e}")
00263             pass
00264         elif e is not None:
00265             print(
00266                 f"ConstructionMonitor/Core.py | Error = {e} | Authentication Error | Please update
keys in AuthUtil")
00267             RESTError(401)
00268             print(e)
00269             pass
00270         else:
00271             pass
00272     except Exception as e:
00273         print(e)
00274         RESTError(1001)
00275         raise SystemExit(1001)
00276

```

Here is the call graph for this function:



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

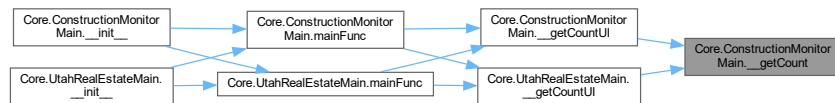
Definition at line 372 of file [ConstructionMonitor/Core.py](#).

```

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
00376         value.
00377         Args:
00378             self: Represent the instance of the class
00379         Returns:
00380             The total number of records in the database
00381         Doc Author:
00382             Willem van der Schans, Trelent AI
00383         """
00384         __count_resp = None
00385         try:
00386             __temp_param_dict = copy.copy(self.__parameterDict)
00387             __count_resp = requests.post(url=self.__restDomain,
00388                                         headers=self.__headerDict,
00389                                         json=__temp_param_dict)
00390         except requests.exceptions.Timeout as e:
00391             print(e)
00392             RESTError(790)
00393             raise SystemExit(790)
00394         except requests.exceptions.TooManyRedirects as e:
00395             print(e)
00396             RESTError(791)
00397             raise SystemExit(791)
00398         except requests.exceptions.MissingSchema as e:
00399             print(e)
00400             RESTError(1101)
00401         except requests.exceptions.RequestException as e:
00402             print(e)
00403             RESTError(405)
00404             raise SystemExit(405)
00405         __count_resp = __count_resp.json()
00406         self.__record_val = __count_resp["hits"]["total"]["value"]
00407         del __count_resp, __temp_param_dict
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417

```

Here is the caller graph for this function:



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

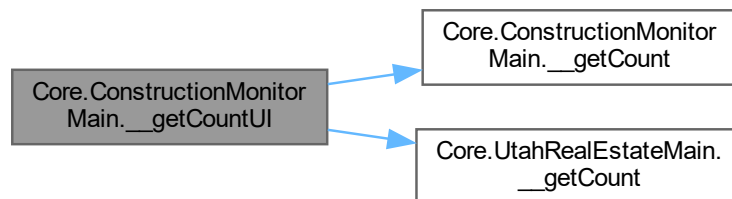
Definition at line 418 of file [ConstructionMonitor/Core.py](#).

```

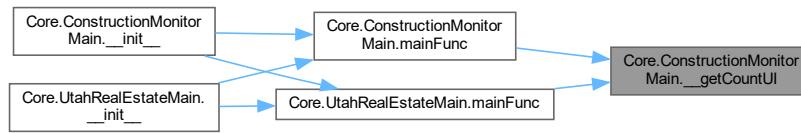
00418     def __getCountUI(self):
00419
00420         """
00421         The __getCountUI function is a wrapper for the __getCount function.
00422         It allows the user to run __getCount in a separate thread, so that they can continue working while it
00423         runs. The function will display a progress bar and update with text as it progresses through its tasks.
00424
00425         Args:
00426             self: Access the class variables and methods
00427
00428         Returns:
00429             The count of the number of records in the database
00430
00431         Doc Author:
00432             Willem van der Schans, Trelent AI
00433         """
00434         if self.__ui_flag:
00435             uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)
00436
00437             threadGui = threading.Thread(target=self.__getCount,
00438                                         daemon=False)
00439             threadGui.start()
00440
00441             while threadGui.is_alive():
00442                 uiObj.textUpdate()
00443                 uiObj.windowPush()
00444             else:
00445                 uiObj.stopWindow()
00446
00447         else:
00448             self.__getCount()

```

Here is the call graph for this function:



Here is the caller graph for this function:



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](#).

```

00333     def __ParameterCreator(self):
00334         """
00335         The __ParameterCreator function is used to create the parameter dictionary that will be passed into
00336         the __Request function. The function takes in a siteClass object and extracts all of its attributes,
00337         except for those that start with '__' or are callable. It then creates a dictionary from these attributes and
00338         stores it as self.__parameterDict.
00339
00340         Args:
00341             self: Make the function a method of the class
00342
00343         Returns:
00344             A dictionary of parameters and a list of non parameter variables
00345
00346         Doc Author:
00347             Willem van der Schans, Trelent AI
00348         """
00349         __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
00350                         not key.startswith('__') and not callable(key)}
00351
00352         self.__restDomain = __Source_dict["rest_domain"]
00353         __Source_dict.pop("rest_domain")
00354         self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
00355         __Source_dict.pop("auth_key")
00356         self.__columnSelection = __Source_dict["SourceInclude"]
00357         __Source_dict.pop("SourceInclude")

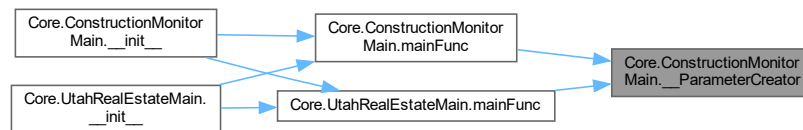
```

```

00358         self.__ui_flag = __Source_dict["ui_flag"]
00359         __Source_dict.pop("ui_flag")
00360         self.__appendFile = __Source_dict["append_file"]
00361         __Source_dict.pop("append_file")
00362
00363         temp_dict = copy.copy(__Source_dict)
00364         for key, value in temp_dict.items():
00365             if value is None:
00366                 __Source_dict.pop(key)
00367             else:
00368                 pass
00369
00370         self.__parameterDict = copy.copy(__Source_dict)
00371

```

Here is the caller graph for this function:



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 __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

Definition at line 277 of file [ConstructionMonitor/Core.py](#).

```

00277     def mainFunc(self):
00278         """
00279         The mainFunc function is the main function of this module. It will be called by the GUI or CLI to
00280         execute the code in this module. The mainFunc function will first create a parameter dictionary using the
00281         __ParameterCreator method, then it will get a count of all records that match its parameters using the __getCountUI
00282         method, and then it will calculate how many batches are needed to retrieve all records with those parameters using
         BatchCalculator.

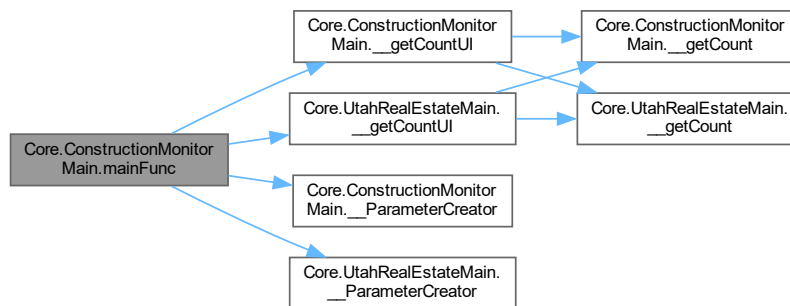
```

```

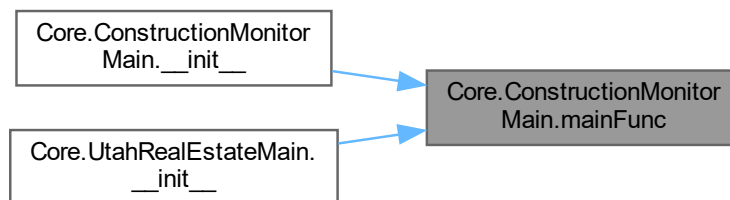
00283     After that it asks if you want to continue with retrieving data from Salesforce (if running in GUI
mode). Then it shows
00284     a progress bar for each
00285
00286     Args:
00287         self: Refer to the current object
00288
00289     Returns:
00290         The dataframe
00291
00292     Doc Author:
00293         Willem van der Schans, Trelent AI
00294     """
00295     self.__ParameterCreator()
00296
00297     print(
00298         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Param Dict =
{self.__parameterDict}")
00299     print(
00300         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Rest Domain =
{self.__restDomain}")
00301
00302     self.__getCountUI()
00303
00304     self.__batches = BatchCalculator(self.__record_val, self.__parameterDict)
00305
00306     print(
00307         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Batches =
{self.__batches} | Rows {self.__record_val}")
00308
00309     if self.__batches != 0:
00310         startTime = datetime.datetime.now().replace(microsecond=0)
00311         eventReturn = BatchInputGui(self.__batches, self.__record_val)
00312         if eventReturn == "Continue":
00313             print(
00314                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Request for
{self.__batches} batches sent to server")
00315             BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00316                                               ParameterDict=self.__parameterDict,
00317                                               HeaderDict=self.__headerDict,
00318                                               ColumnSelection=self.__columnSelection,
00319                                               BatchesNum=self.__batches,
00320                                               Type="construction_monitor")
00321             BatchGuiObject.BatchGuiShow()
00322             self.dataframe = BatchGuiObject.dataframe
00323             print(
00324                 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()))}")
00325             FileSaver("cm", self.dataframe, self.__appendFile)
00326         else:
00327             print(
00328                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Request for
{self.__batches} batches canceled by user")
00329         else:
00330             RESTError(994)
00331             raise SystemExit(994)
00332

```

Here is the call graph for this function:



Here is the caller graph for this function:



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](#).

```
00006     def __init__(self):
00007         """
00008         The __init__ function is called when the class is instantiated.
00009         It sets the initial value of self.__value to 0.
00010
00011         Args:
00012             self: Represent the instance of the class
00013
00014         Returns:
00015             Nothing
00016
00017         Doc Author:
00018             Willem van der Schans, Trelent AI
00019         """
00020         self.__value = 0
00021
```

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](#).

```
00039     def getValue(self):
00040         """
00041         The getValue function returns the value of the private variable __value.
00042         This is a getter function that allows access to this private variable.
00043
00044         Args:
00045             self: Represent the instance of the class
00046
00047         Returns:
00048             The value of the instance variable
00049
00050         Doc Author:
00051             Willem van der Schans, Trelent AI
00052         """
00053         return self.__value
00054
```

Here is the caller graph for this function:



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](#).

```
00022     def setValue(self, value):
00023         """
00024         The setValue function sets the value of the object.
00025
00026
00027         Args:
00028             self: Represent the instance of the class
00029             value: Set the value of the instance variable __value
00030
00031         Returns:
00032             The value that was passed to it
00033
00034         Doc Author:
00035             Willem van der Schans, Trelent AI
00036         """
00037         self.__value = value
00038
```

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](#).

```
00055     def whileValue(self):
00056         """
00057         The whileValue function is a function that will run the getValue function until it is told to stop.
00058         This allows for the program to constantly be checking for new values from the sensor.
00059
00060         Args:
00061             self: Refer to the current instance of the class
00062
00063         Returns:
00064             The value of the input
00065
00066         Doc Author:
00067             Willem van der Schans, Trelent AI
```

```
00068         """
00069         while True:
00070             self.getValue()
```

Here is the call graph for this function:



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 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

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
00019         in this class.
00020         The __init__ function takes two arguments: self and method. The first argument, self, refers to an
00021         instance of a
00022         class (in this case it's an instance of DataFrameSaver). The second argument, method refers to a
00023         string value that
00024         is passed into DataFrameSaver when it's instantiated.
00025
00026         Args:
00027         self: Represent the instance of the class
00028         method: Determine which dataframe to append the new data to
00029         outputDF: Pass in the dataframe that will be saved to a csv file
00030         AppendingPath: Specify the path to an existing csv file that you want to append your dataframe to
00031
00032         Returns:
00033         Nothing
00034
00035         Doc Author:
00036         Willem van der Schans, Trelent AI
00037         """
00038         self.docPath = Path(os.path.expanduser('~/.Documents')).joinpath("GardnerUtilData").joinpath(
00039             datetime.datetime.today().strftime('%m%d%Y'))
00040         self.data = outputDF
```

```

00038     self.dataAppending = None
00039     self.appendFlag = True
00040     self.fileName = f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
00041     self.uiFlag = True
00042
00043     if method.lower() == "ure":
00044         self.primaryKey = "ListingKeyNumeric"
00045     elif method.lower() == "cm":
00046         self.primaryKey = "id"
00047     elif "realtor" in method.lower():
00048         self.primaryKey = None
00049         self.uiFlag = False
00050     elif method.lower() == "cfbp":
00051         self.primaryKey = None
00052         self.uiFlag = False
00053     else:
00054         raise ValueError("method input is invalid choice one of 4 options: URE, CM, Realtor, CFBP")
00055
00056     if AppendingPath is None:
00057         self.appendFlag = False
00058     else:
00059         self.dataAppending = pd.read_csv(AppendingPath)
00060
00061     if self.appendFlag:
00062         if self.primaryKey is not None:
00063             # Due to low_memory loading the columns are not typed properly,
00064             # since we are comparing this will be an issue since we need to do type comparisons,
00065             # so here we coerce the types of the primary keys to numeric.
00066             # If another primary key is ever chosen make sure to core to the right data type.
00067             self.dataAppending[self.primaryKey] = pd.to_numeric(self.dataAppending[self.primaryKey])
00068             self.data[self.primaryKey] = pd.to_numeric(self.data[self.primaryKey])
00069
00070             self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(subset=[self.primaryKey],
                                                                    keep="last")
00071         else:
00072             self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(keep="last")
00073     else:
00074         self.outputFrame = self.data
00075
00076     if os.path.exists(self.docPath):
00077         self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
00078     else:
00079         os.mkdir(self.docPath)
00080         self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
00081
00082     if self.uiFlag:
00083         if self.appendFlag:
00084             PopupWrapped(text=f"File Appended and Saved to {self.docPath.joinpath(self.fileName)}",
00085                          windowType="savedLarge")
00086
00087             # Logging
00088             print(
00089                 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")
00090             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]}")
00091         else:
00092             PopupWrapped(text=f"File Saved to {self.docPath.joinpath(self.fileName)}",
00093                          windowType="savedLarge")
00094
00095             # Logging
00096             print(
00097                 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")
00098     else:
00099         pass
00100

```

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](#).

```
00101     def getPath(self):
00102         """
00103         The getPath function returns the path to the file.
00104         It is a string, and it joins the docPath with the fileName.
00105
00106         Args:
00107             self: Represent the instance of the class
00108
00109         Returns:
00110             The path to the file
00111
00112         Doc Author:
00113             Willem van der Schans, Trelent AI
00114         """
00115         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

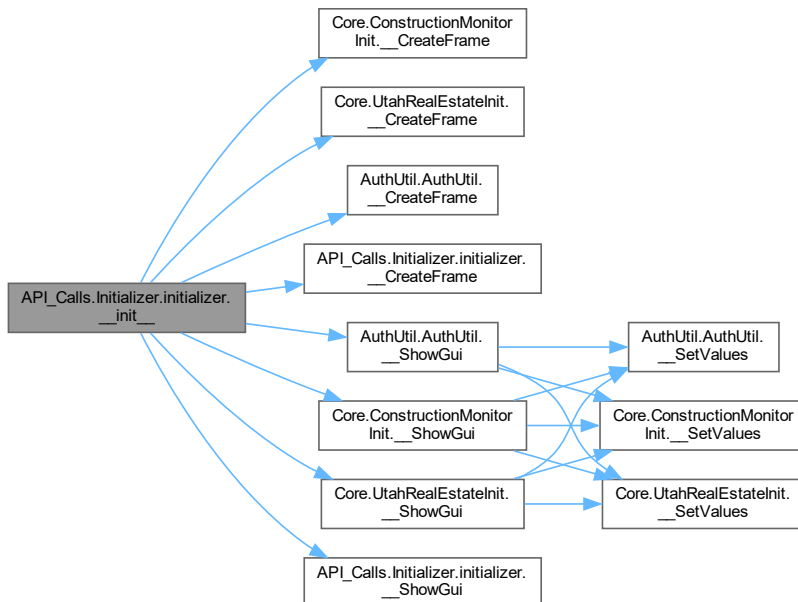
Definition at line 24 of file [Initializer.py](#).

```

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:



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

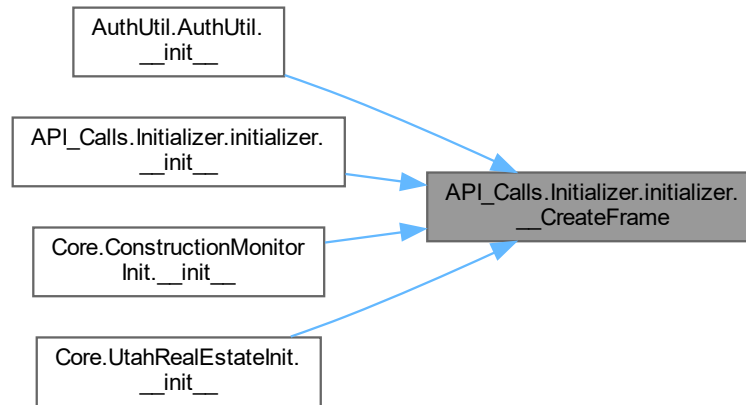
Doc Author:

Willem van der Schans, Trelent AI

Definition at line 136 of file [Initializer.py](#).

```
00136     def __CreateFrame(self):
00137
00138         """
00139         The __CreateFrame function is a helper function that creates the layout for the main window.
00140         It returns a list of lists, which is then passed to sg.Window() as its layout parameter.
00141
00142         Args:
00143             self: Represent the instance of the class
00144
00145         Returns:
00146             A list of lists, which is then passed to the sg
00147
00148         Doc Author:
00149             Willem van der Schans, Trelent AI
00150         """
00151         sg.theme('Default1')
00152
00153         line0 = [sg.HSeparator()]
00154
00155         line1 = [sg.Image(ImageLoader("logo.png")),
00156                 sg.Push(),
00157                 sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"), justification="center"),
00158                 sg.Push(),
00159                 sg.Push()]
00160
00161         line3 = [sg.HSeparator()]
00162
00163         line4 = [sg.Push(),
00164                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"), justification="center"),
00165                 sg.Push()]
00166
00167         line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)), sg.Push(),
00168                 sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00169
00170         line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(),
00171                 sg.Button("CFPB Mortgage", size=(20, None)),
00172                 sg.Push()]]
00173
00174         line8 = [sg.HSeparator()]
00175
00176         line9 = [sg.Push(),
00177                 sg.Text("Utilities", font=("Helvetica", 10, "bold"), justification="center"),
00178                 sg.Push()]
00179
00180         line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
00181                 sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00182
00183         line11 = [sg.HSeparator()]
00184
00185         layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10, line11]
00186
00187         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., `"Text"`, `"InputText"`, 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.

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:
```

```

00056         1) The type of element to be displayed (e.g., "Text", "InputText", etc.)
00057         2) A dictionary containing any additional parameters for that element (e.g., size, default value,
etc.)
00058         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
00059
00060     Args:
00061         self: Represent the instance of the class
00062         layout: Pass the layout of the window to be created
00063         text: Set the title of the window
00064
00065     Returns:
00066         A window object
00067
00068     Doc Author:
00069         Willem van der Schans, Trelent AI
00070     """
00071     # Todo Gitlab Update
00072     versionChecker()
00073
00074     window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00075                       finalize=True,
00076                       icon=ImageLoader("taskbar_icon.ico"))
00077
00078     while True:
00079         event, values = window.read()
00080
00081         if event == "Construction Monitor":
00082             print(
00083                 f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Initiating Construction Monitor API Call-----")
00084             ConstructionMonitorMain(ConstructionMonitorInit())
00085             print(
00086                 f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing Construction Monitor API Call-----\n")
00087             elif event == "Utah Real Estate":
00088                 print(
00089                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Initiating Utah Real Estate API Call-----")
00090                 UtahRealEstateMain(UtahRealEstateInit())
00091                 print(
00092                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing Utah Real Estate API Call-----\n")
00093             elif event == "Realtor.Com":
00094                 print(
00095                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Initiating Realtor.com API Call-----")
00096                 realtorCom()
00097                 print(
00098                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing Realtor.com API Call-----\n")
00099             elif event == "CFPB Mortgage":
00100                 print(
00101                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Initiating ffiec.cfpb API Call-----")
00102                 CFBP()
00103                 print(
00104                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing ffiec.cfpb API Call-----\n")
00105             elif event == "Authorization Utility":
00106                 print(
00107                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Initiating Authorization Utility-----")
00108                 AuthUtil()
00109                 print(
00110                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing Authorization Utility-----\n")
00111             elif event == "Open Data Folder":
00112                 print(
00113                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Data Folder Opened-----")
00114                 try:
00115                     os.system(f"start
{Path(os.path.expanduser('~\\Documents')).joinpath('GardnerUtilData')}")
00116                 except:
00117                     try:
00118                         os.system(f"start {Path(os.path.expanduser('~\\Documents'))}")
00119                     except Exception as e:
00120                         print(f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
Initializer.py | Error = {e} | Documents folder not found")
00121                 PopupWrapped(

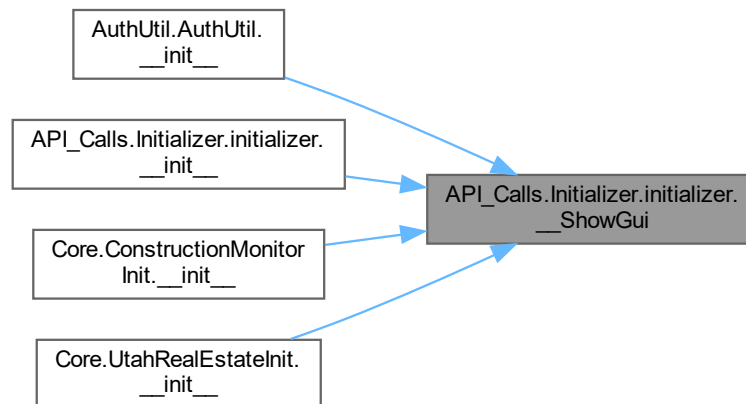
```

```

00122             text="Documents folder not found. Please create a Windows recognized documents
00123 folder",
00124             windowType="errorLarge")
00125         elif event in ('Exit', None):
00126             try:
00127                 break
00128             except Exception as e:
00129                 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.")
00130                 break
00131             elif event == sg.WIN_CLOSED or event == "Quit":
00132                 break
00133
00134         window.close()
00135

```

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:



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 information.

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](#).

```

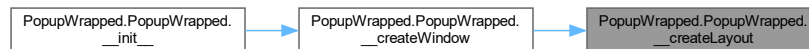
00058     def __createLayout(self):
00059         """
00060         The __createLayout function is used to create the layout of the window.
00061         The function takes class variables and returns a window layout.
00062         It uses a series of if statements to determine what type of window it is, then creates a layout based
on that information.
00063         Args:
00064             self: Refer to the current instance of a class
00065         Returns:
00066             A list of lists
00067         Doc Author:
00068             Willem van der Schans, Trelent AI
00069         """
00070         sg.theme('Default1')
00071         __Line1 = None
00072         __Line2 = None
00073
00074         if self.__type == "notice":
00075             __Line1 = [sg.Push(),
00076                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
00077                        sg.Text(self.__text, justification="center", key="-textField-", sg.Push())
00078             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00079         elif self.__type == "noticeLarge":
00080             __Line1 = [sg.Push(),
00081                        sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
00082                        sg.Text(self.__text, justification="center", key="-textField-", sg.Push())
00083             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00084         elif self.__type == "savedLarge":
00085             if self.__errorFlag:
00086                 __Line1 = [sg.Push(),
00087                            sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
```

```

00088         sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00089         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00090     else:
00091         __Line1 = [sg.Push(),
00092             sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
00093             sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00094         __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)), sg.Ok(focus=True, size=(10,
00095 1)), sg.Push()]
00096     elif self.__type == "errorLarge":
00097         __Line1 = [sg.Push(),
00098             sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00099             sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00100         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00101     elif self.__type == "FatalErrorLarge":
00102         __Line1 = [sg.Push(),
00103             sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00104             sg.Text(self.__text, justification="left", key="-textField-"), sg.Push()]
00105         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00106     elif self.__type == "error":
00107         __Line1 = [sg.Push(),
00108             sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00109             sg.Text(f"{self.__text}: {self.__error}", justification="center",
00110 key="-textField-"),
00111             sg.Push()]
00112         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00113     elif self.__type == "AuthError":
00114         __Line1 = [sg.Push(),
00115             sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00116             sg.Text(f"{self.__text}", justification="center", key="-textField-"),
00117             sg.Push()]
00118         __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool [Web Browser]",
00119             sg.Ok(button_text="Return", focus=True, size=(10, 1)), sg.Push()]
00120     elif self.__type == "versionWindow":
00121         __Line1 = [sg.Push(),
00122             sg.Text(f"{self.__text}", justification="left", key="-textField-"),
00123             sg.Push()]
00124         __Line2 = [sg.Push(), sg.Button(button_text="Download"),
00125             sg.Ok(button_text="Continue", focus=True, size=(10, 1)), sg.Push()]
00126     elif self.__type == "progress":
00127         __Line1 = [sg.Push(),
00128             sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00129     if self.__type == "progress":
00130         self.__layout = [__Line1, ]
00131     else:
00132         self.__layout = [__Line1, __Line2]

```

Here is the caller graph for this function:



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 the

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](#).

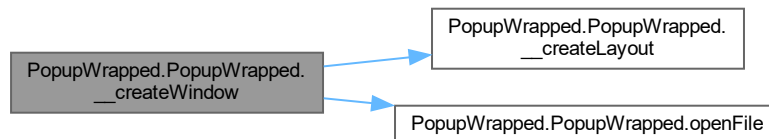
```
00133     def __createWindow(self):
00134         """
00135         The __createWindow function is used to create the window object that will be displayed.
00136         The function takes class variables and a window object. The function first calls __createLayout, which
00137         creates the layout for the window based on what type of message it is (error, notice, progress). Then it
00138         uses PySimpleGUI's Window class to create a new window with that layout and some other parameters such as
00139         title and icon. If this is not a progress bar or permanent message then we start a timer loop that waits
00140         until either 100 iterations have passed or an event has been triggered (such as clicking &quot;Ok&quot; or
00141         closing the window). Once one of these events occurs
00142
00143         Args:
00144             self: Reference the instance of the class
00145         Returns:
00146             A window object
00147         Doc Author:
00148             Willem van der Schans, Trelent AI
00149         """
00150         self.__createLayout()
00151
00152         if self.__type == "progress":
00153             self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout,
00154                                         finalize=True,
00155                                         modal=True,
00156                                         keep_on_top=True,
00157                                         disable_close=False,
00158                                         icon=ImageLoader("taskbar_icon.ico"),
00159                                         size=(290, 50))
00160
00161         elif self.__type == "noticeLarge":
00162             self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
00163                                         modal=True,
00164                                         keep_on_top=True,
00165                                         disable_close=False,
00166                                         icon=ImageLoader("taskbar_icon.ico"))
00167
00168         elif self.__type == "savedLarge":
00169             self.__windowObj = sg.Window(title="Notice", layout=self.__layout, finalize=True,
00170                                         modal=True,
00171                                         keep_on_top=False,
00172                                         disable_close=False,
00173                                         icon=ImageLoader("taskbar_icon.ico"))
00174
00175         elif self.__type == "errorLarge":
00176             self.__windowObj = sg.Window(title="Error", layout=self.__layout, finalize=True,
00177                                         modal=True,
00178                                         keep_on_top=True,
00179                                         disable_close=False,
00180                                         icon=ImageLoader("taskbar_icon.ico"))
00181
00182         elif self.__type == "FatalErrorLarge":
00183             self.__windowObj = sg.Window(title="Fatal Error", layout=self.__layout, finalize=True,
00184                                         modal=True,
00185                                         keep_on_top=True,
00186                                         disable_close=False,
00187                                         icon=ImageLoader("taskbar_icon.ico"))
00188
00189         elif self.__type == "AuthError":
00190             self.__windowObj = sg.Window(title="Authentication Error", layout=self.__layout,
00191                                         finalize=True,
00192                                         modal=True,
00193                                         keep_on_top=True,
00194                                         disable_close=False,
00195                                         icon=ImageLoader("taskbar_icon.ico"))
00196
00197         elif self.__type == "versionWindow":
00198             self.__windowObj = sg.Window(title="Update Notice", layout=self.__layout, finalize=True,
00199                                         modal=True,
00200                                         keep_on_top=True,
00201                                         disable_close=False,
00202                                         icon=ImageLoader("taskbar_icon.ico"))
00203
00204         else:
00205             self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout,
00206                                         finalize=True,
00207                                         modal=True,
00208                                         keep_on_top=True,
00209                                         disable_close=False,
00210                                         icon=ImageLoader("taskbar_icon.ico"),
```

```

00195                                     size=(290, 80))
00196
00197     if self.__type != "progress" or self.__type.startswith("perm"):
00198         timer = 0
00199         while timer < 100:
00200             event, values = self.__windowObj.read()
00201             if event == "Ok" or event == sg.WIN_CLOSED or event == "Return" or event == "Continue":
00202                 break
00203             elif event == "Open Generation Tool [Web Browser]":
00204                 webbrowser.open(settings.settingGenerationToolLink, new=2, autoraise=True)
00205                 pass
00206             elif event == "Open File":
00207                 threadFile = threading.Thread(target=self.openFile,
00208                                                daemon=False)
00209                 threadFile.start()
00210                 time.sleep(3)
00211                 break
00212             elif event == "Download":
00213                 # Todo Gitlab Update
00214                 webbrowser.open(settings.settingDownloadSourceLink, new=2,
00215                                autoraise=True)
00216                 pass
00217                 time.sleep(0.1)
00218
00219         if self.__type == "FatalErrorLarge":
00220             try:
00221                 os.system(
00222                     f"start
00223 {Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')})"
00224             except Exception as e:
00225                 print(
00226                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
00227 PopupWrapped.py | Error = {e} | Log Folder not found please search manually for
00228 %APPDATA%\Roaming\GardnerUtil\Logs\n")
00226
00227         self.__windowObj.close()
00228

```

Here is the call graph for this function:



Here is the caller graph for this function:



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

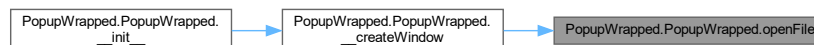
Doc Author:

Willem van der Schans, Trelent AI

Definition at line 290 of file [PopupWrapped.py](#).

```
00290     def openFile(self):
00291         """
00292         The openFile function opens the file that is associated with the
00293         document object. It does this by calling os.system and passing it
00294         self.__docpath as an argument.
00295
00296         Args:
00297             self: Represent the instance of the object itself
00298
00299         Returns:
00300             The filepath of the document
00301
00302         Doc Author:
00303             Willem van der Schans, Trelent AI
00304         """
00305         os.system(self.__docpath)
```

Here is the caller graph for this function:



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](#).

```
00229     def stopWindow(self):
00230         """
00231         The stopWindow function is used to close the window object that was created in the startWindow
function.
00232         This is done by calling the close() method on self.__windowObj, which will cause it to be destroyed.
00233         Args:
00234             self: Represent the instance of the class
00235         Returns:
00236             The window object
00237         Doc Author:
00238             Willem van der Schans, Trelent AI
00239         """
00240         self.__windowObj.close()
00241
```

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](#).

```
00242     def textUpdate(self, sleep=0.5):
00243         """
00244         The textUpdate function is a function that updates the text in the text field.
00245         It does this by adding dots to the end of it, and then removing them. This creates
00246         a loading effect for when something is being processed.
00247         Args:
00248             self: Refer to the object itself
00249             sleep: Control the speed of the text update
00250         Returns:
00251             A string that is the current text of the text field
00252         Doc Author:
00253             Willem van der Schans, Trelent AI
00254         """
00255         self.__counter += 1
00256         if self.__counter == 4:
00257             self.__counter = 1
00258         newString = ""
00259         if self.__type == "notice":
00260             pass
00261         elif self.__type == "error":
00262             pass
00263         elif self.__type == "progress":
00264             newString = f"{self.__text}{'.' * self.__counter}"
00265         self.__windowObj.write_event_value('update-textField-', newString)
00266
00267         time.sleep(sleep)
00268
```


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](#).

```
00269     def windowPush(self):
00270
00271         """
00272         The windowPush function is used to update the values of a window object.
00273         The function takes in an event and values from the window object, then checks if the event starts
00274         with 'update'.
00275         If it does, it will take everything after 'update' as a key for updating that specific value.
00276         It will then update that value using its key and refresh the window.
00277         Args:
00278         self: Reference the object that is calling the function
00279         Returns:
00280         A tuple containing the event and values
00281         Doc Author:
00282         Willem van der Schans, Trelent AI
00283         """
00284         event, values = self.__windowObj.read()
00285
00286         if event.startswith('update'):
00287             __key_to_update = event[len('update'):]
00288             self.__windowObj[__key_to_update].update(values[event])
00289             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

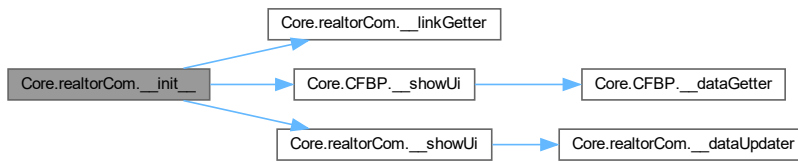
Returns:
A new object

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
00022         anything else in your class.
00023
00024         Args:
00025             self: Represent the instance of the class
00026
00027         Returns:
00028             A new object
00029
00030         Doc Author:
00031             Willem van der Schans, Trelent AI
00032         """
00033         self.__page_html = None
00034         self.__update_date = None
00035         self.__last_date = None
00036         self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00037         self.__linkDict = {}
00038         self.dfState = None
00039         self.dfCounty = None
00040         self.dfZip = None
00041         self.uiString = "Files Saved to \n"
00042
00043         eventReturn = confirmDialog()
00044         if eventReturn == "Continue":
00045             page_html = requests.get(settings.settingRealtorLink).text
00046             self.__page_html = BeautifulSoup(page_html, "html.parser")
00047             startTime = datetime.datetime.now().replace(microsecond=0)
00048             self.__linkGetter()
00049             print(
00050                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Link Dictionary =
00051                 {self.__idDict}")
00052             self.__showUi()
00053             PopupWrapped(text=self.uiString, windowType="noticeLarge")
00054             print(
00055                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Data retrieved with
00056                 in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) -
00057                 startTime).total_seconds()))}")
00058             else:
00059                 print(
00060                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | User Canceled
00061                     Request")
00062             pass
```

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](#).

```

00114     def __dataUpdater(self):
00115
00116         """
00117         The __dataUpdater function is a private function that updates the dataframes for each of the three
00118         types of realtor data. It takes class variables and return the path to the saved file. The
00119         function first creates an empty
00119         dictionary called tempdf, then iterates through each key in self.__idDict (which contains all
00120         three ids).
00120         For each key, it reads in a csv file from the link associated with that id and saves it to tempdf
00121         as a pandas
00121         DataFrame object. Then, depending on which type of realtor data we are dealing with
00122         (State/County/Zip), we save
00122
00123
00124         Args:
00125             self: Access the attributes and methods of the class
00126
00127         Returns:
00128             The path of the saved file
00129
00129
```

```

00130     Doc Author:
00131         Willem van der Schans, Trelent AI
00132     """
00133     for key, value in self.__idDict.items():
00134         tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00135
00136         if key == "State":
00137             self.dfState = tempdf
00138         elif key == "County":
00139             self.dfCounty = tempdf
00140         elif key == "Zip":
00141             self.dfZip = tempdf
00142
00143     FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00144     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](#).

```

00087     def __linkGetter(self):
00088
00089         """
00090         The __linkGetter function is a private function that takes the idDict dictionary and adds
00091         a link to each entry in the dictionary. The link is used to access historical data for each
00092         scope symbol.
00093
00094         Args:
00095             self: Refer to the object itself
00096
00097         Returns:
00098             A dictionary of all the links to the history pages
00099
00100         Doc Author:

```

```

00101         Willem van der Schans, Trelent AI
00102         """
00103         for key, value in self.__idDict.items():
00104             for row in self.__page_html.find_all("div", {"class": "monthly"}):
00105                 try:
00106                     for nestedRow in row.find_all("a"):
00107                         if "History" in str(nestedRow.get("href")) and key in str(nestedRow.get("href")):
00108                             self.__idDict[key] = {"id": value, "link": nestedRow.get("href")}
00109                 except Exception as e:
00110                     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")
00111                     RESTError(801)
00112                     raise SystemExit(801)
00113

```

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](#).

```

00059     def __showUi(self):
00060
00061         """
00062         The __showUi function is a helper function that creates and displays the progress window.
00063         It also starts the dataUpdater thread, which will update the progress bar as it runs.
00064
00065
00066         Args:
00067             self: Represent the instance of the class
00068
00069         Returns:
00070             A popupwrapped object
00071
00072         Doc Author:

```

```

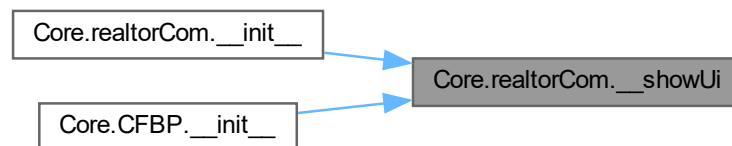
00073     Willem van der Schans, Trelent AI
00074     """
00075     uiObj = PopupWrapped(text="Request running", windowType="progress", error=None)
00076
00077     threadGui = threading.Thread(target=self.__dataUpdater,
00078                                 daemon=False)
00079     threadGui.start()
00080
00081     while threadGui.is_alive():
00082         uiObj.textUpdate()
00083         uiObj.windowPush()
00084     else:
00085         uiObj.stopWindow()
00086

```

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.cfbp.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](#).

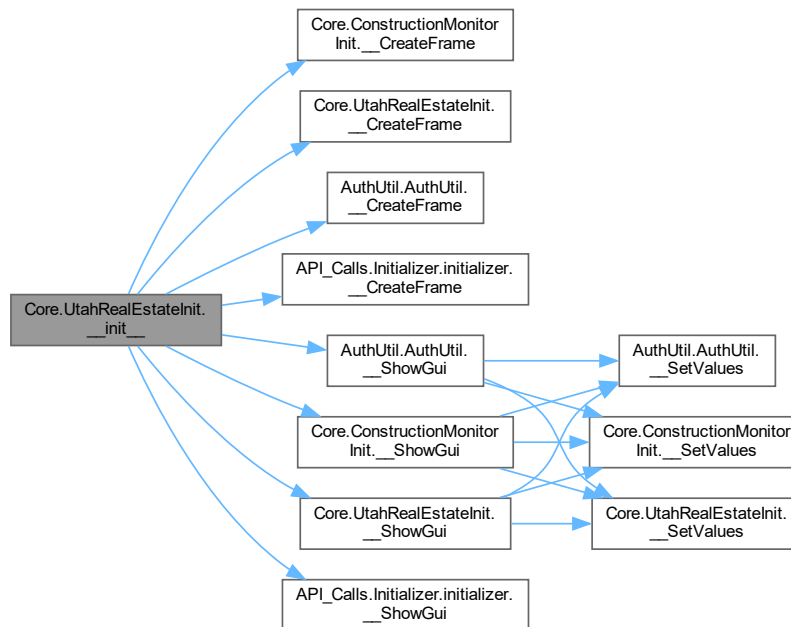
```
00027     def __init__(self):
00028
00029         """
00030         The __init__ function is called when the class is instantiated.
00031         It sets up the initial state of the object.
00032
00033         Args:
00034             self: Represent the instance of the class
00035
00036         Returns:
00037             The __createframe function
00038
00039         Doc Author:
00040             Willem van der Schans, Trelent AI
00041         """
00042
00043         self.StandardStatus = None
00044         self.ListedOrModified = None
00045         self.dateStart = None
```

```

00046         self.dateEnd = None
00047         self.select = None
00048         self.file_name = None
00049         self.append_file = None
00050
00051         self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")
00052

```

Here is the call graph for this function:



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

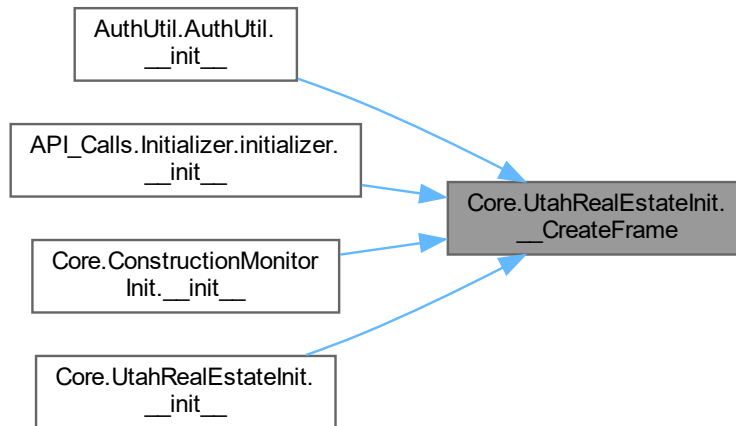
Definition at line 93 of file [UtahRealEstate/Core.py](#).

```

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"),
00114                        justification="center"),
00115                 sg.Push(),
00116                 sg.Push()]
00117
00118         line1 = [sg.HSeparator()]
00119
00120         line2 = [sg.Text("MLS Status : ", size=(15, None), justification="Right"),
00121                 sg.DropDown(default_value="Active", values=["Active", "Closed"], key="-status-",
00122                        size=(31, 1))]
00123
00124         line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00125                 sg.DropDown(default_value="Listing Date", values=["Listing Date", "Modification Date",
00126                        "Close Date"],
00127                        key="-type-", size=(31, 1))]
00128
00129         line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00130                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"),
00131                        key="-DateStart-",
00132                        disabled=False, size=(20, 1)),
00133                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
00134                        target="-DateStart-")]
00135
00136         line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00137                 sg.Input(default_text=(date.today()).strftime("%Y-%m-%d"), key="-DateEnd-",
00138                        disabled=False,
00139                        size=(20, 1)),
00140                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-end_date-',
00141                        target="-DateEnd-")]
00142
00143         line7 = [sg.HSeparator()]
00144
00145         line8 = [sg.Push(),
00146                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
00147                 sg.Push()]
00148
00149         line9 = [sg.HSeparator()]
00150
00151         line10 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
00152                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
00153                        size=(20, 1)),
00154                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
00155                        target="-AppendingFile-")]
00156
00157         line11 = [sg.HSeparator()]
00158
00159         line12 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00160
00161         layout = [line00, line0, line1, line2, line3, line4, line5, line7, line8, line9, line10, line11,
00162                 line12]
00163
00164         return layout

```

Here is the caller graph for this function:



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](#).

```
00159     def __SetValues(self, values):
00160
00161         """
00162         The __SetValues function is used to set the values of the variables that are used in the
00163         __GetData function. The values are passed from a dictionary called 'values' which is created
00164         by parsing through an XML file using ElementTree. This function also sets default values for
00165         some of these variables if they were not specified in the XML file.
00166
00167         Args:
```

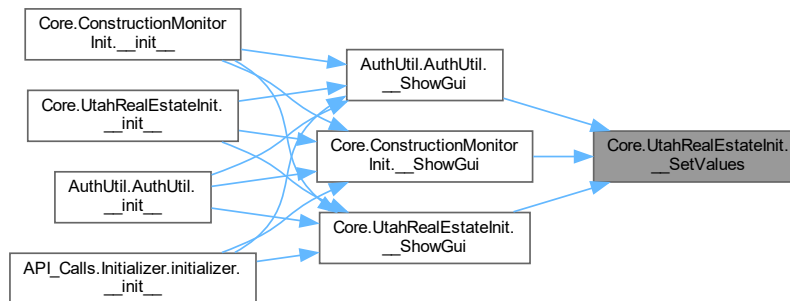


```

00168         self: Represent the instance of the class
00169         values: Pass the values from the gui to this function
00170
00171     Returns:
00172         A dictionary with the following keys:
00173
00174     Doc Author:
00175         Willem van der Schans, Trelent AI
00176     """
00177     self.StandardStatus = values["-status-"]
00178
00179     self.ListedOrModified = values["-type-"]
00180
00181     if values["-DateStart-"] != "":
00182         self.dateStart = values["-DateStart-"]
00183     else:
00184         self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")
00185
00186     if values["-DateEnd-"] != "":
00187         self.dateEnd = values["-DateEnd-"]
00188     else:
00189         self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00190
00191     self.select = None
00192
00193     if values["-append_file-"] != "":
00194         self.append_file = str(values["-append_file-"])
00195     else:
00196         self.append_file = None
00197
00198

```

Here is the caller graph for this function:



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

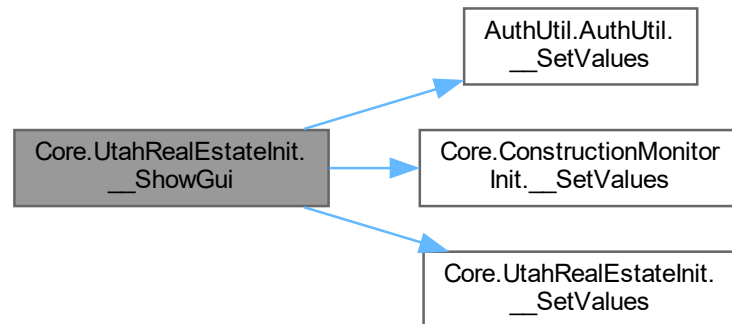
Doc Author:

Willem van der Schans, Trelent AI

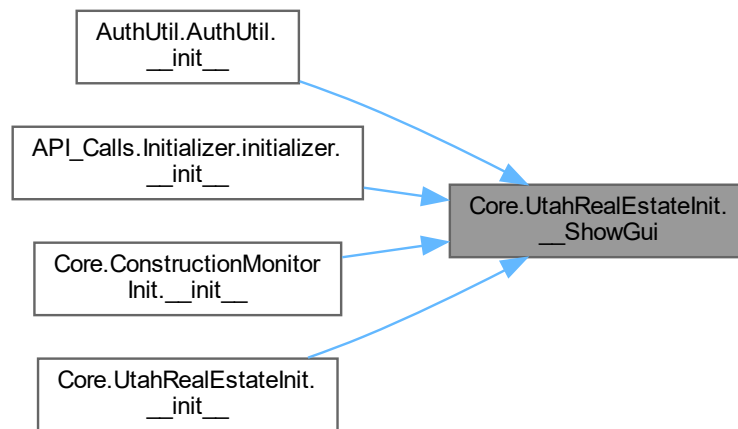
Definition at line 53 of file `UtahRealEstate/Core.py`.

```
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
00059         __ShowGui
00059         method then uses these parameters to create an instance of sg.Window with all its attributes set
00060         accordingly.
00061
00062         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:
00066             A dictionary of values
00067
00068         Doc Author:
00069             Willem van der Schans, Trelent AI
00070
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

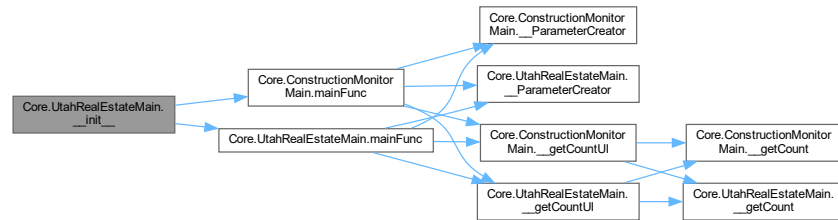
Doc Author:

Willem van der Schans, Trelent AI

Definition at line 201 of file [UtahRealEstate/Core.py](#).

```
00201     def __init__(self, siteClass):
00202
00203         """
00204         The __init__ function is the first function that runs when an object of this class is created.
00205         It sets up all the variables and functions needed for this class to work properly.
00206
00207         Args:
00208             self: Represent the instance of the class
00209             siteClass: Determine which site to pull data from
00210
00211         Returns:
00212             Nothing
00213
00214         Doc Author:
00215             Willem van der Schans, Trelent AI
00216         """
00217         self.dataframe = None
00218         self.__batches = 0
00219         self.__siteClass = siteClass
00220         self.__headerDict = None
00221         self.__parameterString = ""
00222         self.__appendFile = None
00223         self.__dateStart = None
00224         self.__dateEnd = None
00225         self.__restDomain = settings.settingURERestDomain
00226         self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00227             "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
00228         self.filePath = Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData").joinpath(
00229             "Security").joinpath("auth.json")
00230         self.key = None
00231         self.__record_val = None
00232
00233         try:
00234             self.mainFunc()
00235         except KeyError as e:
00236             # This allows for user cancellation of the program using the quit button
00237             if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
00238                 RESTError(1101)
00239                 print(e)
00240                 pass
00241             else:
00242                 pass
00243         except Exception as e:
00244             print(e)
00245             RESTError(1001)
00246             raise SystemExit(1001)
00247
```

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](#).

```

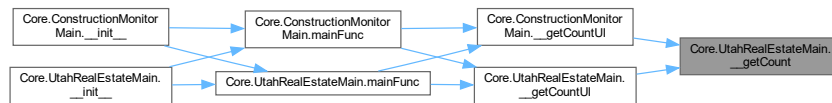
00366     def __getCount(self):
00367         """
00368         The __getCount function is used to determine the number of records that will be returned by the query.
00369         This function is called when a user calls the count() method on a ReST object. The __getCount function
00370         uses
00371         the $count parameter in OData to return only an integer value representing how many records would be
00372         returned
00373         by the query.
00374         Args:
00375         self: Represent the instance of the class
00376         Returns:
00377         The number of records in the data set
00378         Doc Author:
00379         Willem van der Schans, Trelent AI
00380         """
00381         __count_resp = None
00382         try:
```

```

00385         __count_resp = requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
00386                                     headers=self.__headerDict)
00387
00388     except requests.exceptions.Timeout as e:
00389         print(e)
00390         RESTError(790)
00391         raise SystemExit(790)
00392     except requests.exceptions.TooManyRedirects as e:
00393         print(e)
00394         RESTError(791)
00395         raise SystemExit(791)
00396     except requests.exceptions.MissingSchema as e:
00397         print(e)
00398         RESTError(1101)
00399     except requests.exceptions.RequestException as e:
00400         print(e)
00401         RESTError(405)
00402         raise SystemExit(405)
00403
00404     self.__record_val = int(__count_resp.json()["@odata.count"])
00405

```

Here is the caller graph for this function:



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](#).

```

00406     def __getCountUI(self):
00407
00408         """
00409         The __getCountUI function is a wrapper for the __getCount function.
00410         It creates a progress window and updates it while the __getCount function runs.
00411         The purpose of this is to keep the GUI responsive while running long processes.
00412
00413         Args:
00414             self: Represent the instance of the class

```

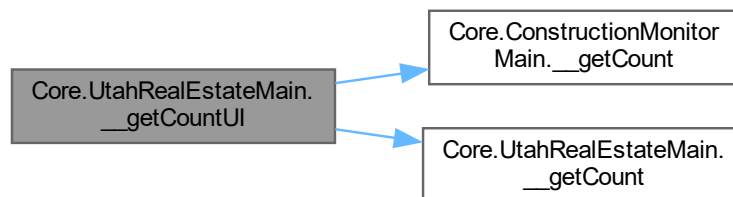


```

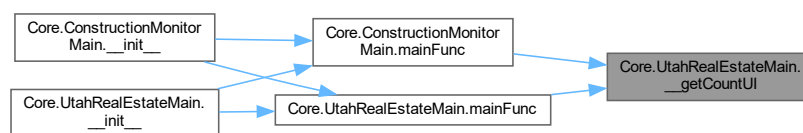
00415
00416 Returns:
00417     A popupwrapped object
00418
00419 Doc Author:
00420     Willem van der Schans, Trelent AI
00421 """
00422 uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)
00423
00424 threadGui = threading.Thread(target=self.__getCount,
00425                             daemon=False)
00426 threadGui.start()
00427
00428 while threadGui.is_alive():
00429     uiObj.textUpdate()
00430     uiObj.windowPush()
00431 else:
00432     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

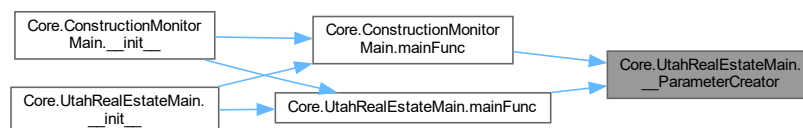
Doc Author:

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 =
00357 f"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContractDate%20le%20{__Source_dict['dateEnd']}"
00358             elif __Source_dict["ListedOrModified"] == "Modification Date":
00359                 filter_string =
00360 f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20and%20ModificationTimestamp%20le%20{__Source_dict['dateEnd']}"
00361             elif __Source_dict["ListedOrModified"] == "Close Date":
00362                 filter_string =
00363 f"$filter=CloseDate%20gt%20{__Source_dict['dateStart']}%20and%20CloseDate%20le%20{__Source_dict['dateEnd']}"
00364
00365         filter_string = filter_string +
00366 f"%20and%20StandardStatus%20has%20data.Models.StandardStatus'{__Source_dict['StandardStatus']}'"
00367
00368         self.__parameterString = filter_string
```

Here is the caller graph for this function:



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 "Run" button in the GUI. The mainFunc function should contain all of your code for running your program. 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

Definition at line 248 of file [UtahRealEstate/Core.py](#).

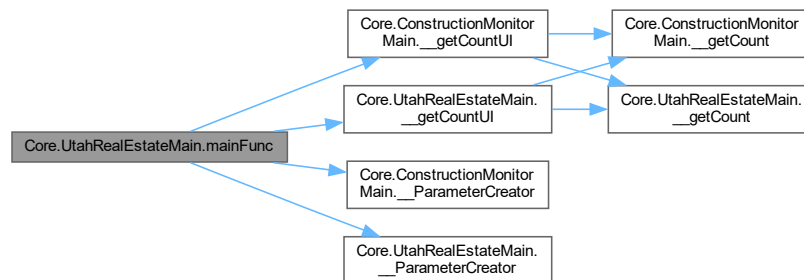
```
00248     def mainFunc(self):
00249
00250         """
00251         The mainFunc function is the main function of this module. It will be called by the GUI when a user
00252         clicks on the "Run" button in the GUI. The mainFunc function should contain all of your code for
00253         running your program, and it should return a dataframe that contains all the data you want to display in your final report.
00254
00255         Args:
00256             self: Reference the object itself
00257
00258         Returns:
00259             A dataframe
00260
00261         Doc Author:
00262             Willem van der Schans, Trelent AI
00263         """
00264         passFlag = False
00265
00266         while not passFlag:
00267             if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
00268                 try:
00269                     f = open(self.keyPath, "rb")
00270                     key = f.readline()
00271                     f.close()
00272                     f = open(self.filePath, "rb")
00273                     authDict = json.load(f)
00274                     fernet = Fernet(key)
00275                     authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
00276                     self.__headerDict = {authDict["ure"]["parameter"]: authkey}
00277                     passFlag = True
00278                 except Exception as e:
00279                     print(
00280                         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
00281                     AuthUtil()
00282             else:
00283                 AuthUtil()
00284
00285         self.__ParameterCreator()
00286
00287         print(
00288             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Param String =
{self.__parameterString}")
00289         print(
00290             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Rest Domain =
{self.__restDomain}")
00291
```

```

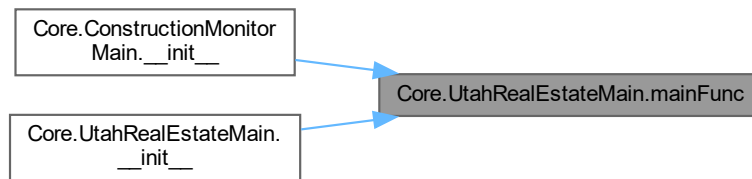
00292         self.__getCountUI()
00293
00294         if self.__record_val is None:
00295             self.__record_val = 0
00296
00297         self.__batches = BatchCalculator(self.__record_val, None)
00298
00299         print(
00300             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Batches =
{self.__batches} | Rows {self.__record_val}")
00301
00302         if self.__batches != 0:
00303             startTime = datetime.datetime.now().replace(microsecond=0)
00304             eventReturn = BatchInputGui(self.__batches, self.__record_val)
00305             if eventReturn == "Continue":
00306                 print(
00307                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Request for
{self.__batches} batches sent to server")
00308                 BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00309                                                    ParameterDict=self.__parameterString,
00310                                                    HeaderDict=self.__headerDict,
00311                                                    BatchesNum=self.__batches,
00312                                                    Type="utah_real_estate")
00313                 BatchGuiObject.BatchGuiShow()
00314                 self.dataframe = BatchGuiObject.dataframe
00315                 print(
00316                     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()))}")
00317                 FileSaver("ure", self.dataframe, self.__appendFile)
00318             else:
00319                 print(
00320                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Request for
{self.__batches} batches canceled by user")
00321             else:
00322                 RESTError(994)
00323                 raise SystemExit(994)
00324

```

Here is the call graph for this function:



Here is the caller graph for this function:



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

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002
00003
00004 from Initializer import initializer
00005
00006 initializer()
```

4.3 AuthUtil.py

```
00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002
00003
00004 import ctypes
00005 import datetime
00006 import json
00007 import os
00008 from pathlib import Path
00009
00010 import PySimpleGUI as sg
00011 from cryptography.fernet import Fernet
00012
00013 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00014 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00015 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00016
00017
00018 class AuthUtil:
00019
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
00025         displays it on screen.
00026
00027         Args:
00028             self: Represent the instance of the class
00029
00030         Returns:
00031             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:
00064             os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))
00065             os.mkdir(self.keyPath)
00066
00067     if os.path.isfile(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w")):
00068         try:
00069             f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "rb")
00070             self.k = f.readline()
00071             f.close()
00072         except Exception as e:
00073             print(e)
00074             RESTError(402)
00075             raise SystemExit(402)
00076     else:
00077         self.k = Fernet.generate_key()
00078         f = open(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), "wb")
00079         f.write(self.k)
00080         f.close()
00081
00082         try:
00083             os.remove(self.filePath.joinpath("auth.json"))
00084         except Exception as e:
00085             # Logging
00086             print(
00087                 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..."")
00088             pass
00089
00090         f = open(self.filePath.joinpath("auth.json"), "wb")
00091         f.close()
00092         self.keyFlag = False
00093
00094     self.__ShowGui(self.__CreateFrame(), "Authenticator Utility")
00095
00096     try:
00097         ctypes.windll.kernel32.SetFileAttributesW(self.keyPath.joinpath("3v45wfvw45wvc4f35.av3ra3rvavcr3w"), 2)
00098     except Exception as e:
00099         # Logging
00100         print(
00101             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..."")
00102         pass
00103
00104     def __SetValues(self, values):
00105
00106         """

```

```

00107     The __SetValues function is called when the user clicks on the "OK" button in the window.
00108     It takes a dictionary of values as an argument, and then uses those values to update
00109     the auth.json file with new keys for both Utah Real Estate and Construction Monitor.
00110
00111     Args:
00112         self: Make the function a method of the class
00113         values: Store the values that are entered into the form
00114
00115     Returns:
00116         A dictionary of the values entered by the user
00117
00118     Doc Author:
00119         Willem van der Schans, Trelent AI
00120     """
00121     ureCurrent = None
00122     cmCurrent = None
00123     keyFile = None
00124     self.popupFlag = False
00125
00126     fernet = Fernet(self.k)
00127
00128     try:
00129         f = open(self.filePath.joinpath("auth.json"), "r")
00130         keyFile = json.load(f)
00131         fileFlag = True
00132     except:
00133         fileFlag = False
00134
00135     # Try initial decoding, if fails pass and write new keys and files
00136     if fileFlag:
00137         try:
00138             ureCurrent = fernet.decrypt(keyFile["ure"]['auth'].decode())
00139         except Exception as e:
00140             # Logging
00141             print(
00142                 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")
00143             ureCurrent = None
00144
00145         try:
00146             cmCurrent = fernet.decrypt(keyFile["cm"]['auth'].decode())
00147         except Exception as e:
00148             # Logging
00149             print(
00150                 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")
00151             cmCurrent = None
00152
00153     if values["-ureAuth-"] != "":
00154         self.jsonDict.update(
00155             {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(values["-ureAuth-"].encode()).decode()}})
00156         self.passFlagUre = True
00157     elif ureCurrent is not None:
00158         self.jsonDict.update(
00159             {"ure": {"parameter": "Authorization", "auth":
fernet.encrypt(ureCurrent.encode()).decode()}})
00160         self.passFlagUre = True
00161     else:
00162         pass
00163
00164     if values["-cmAuth-"] != "":
00165         if values["-cmAuth-"].startswith("Basic"):
00166             self.jsonDict.update(
00167                 {"cm": {"parameter": "Authorization",
00168                     "auth": fernet.encrypt(values["-cmAuth-"].encode()).decode()}})
00169             self.passFlagCm = True
00170     else:
00171         PopupWrapped("Please make sure you provide a HTTP Basic Auth key for construction
Monitor",
00172                     windowType="AuthError")
00173         self.popupFlag = True
00174         pass
00175     elif ureCurrent is not None:
00176         self.jsonDict.update(
00177             {"cm": {"parameter": "Authorization", "auth":
fernet.encrypt(cmCurrent.encode()).decode()}})
00178         self.passFlagUre = True
00179     else:

```

```

00180         pass
00181
00182         if not self.passFlagUre and not self.passFlagCm:
00183             PopupWrapped("Please make sure you provide keys for both Utah Real estate and Construction
Monitor",
                           windowType="errorLarge")
00184
00185         if self.passFlagCm and not self.passFlagUre:
00186             PopupWrapped("Please make sure you provide a key for Utah Real estate",
                           windowType="errorLarge")
00187
00188         if not self.passFlagCm and self.passFlagUre and not self.popupFlag:
00189             PopupWrapped("Please make sure you provide a key for Construction Monitor",
                           windowType="errorLarge")
00189
00190         if self.popupFlag:
00191             pass
00192         else:
00193             jsonOut = json.dumps(self.jsonDict, indent=4)
00194             f = open(self.filePath.joinpath("auth.json"), "w")
00195             f.write(jsonOut)
00196
00197     def __ShowGui(self, layout, text):
00198         """
00199         The __ShowGui function is a helper function that displays the GUI to the user.
00200         It takes in two arguments: layout and text. The layout argument is a list of lists,
00201         which contains all the elements that will be displayed on screen. The text argument
00202         is simply what will be displayed at the top of the window.
00203
00204         Args:
00205             self: Represent the instance of the class
00206             layout: Pass the layout of the gui to be displayed
00207             text: Set the title of the window
00208
00209         Returns:
00210             A window object
00211
00212         """
00213         window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00214                             finalize=True,
00215                             icon=ImageLoader("taskbar_icon.ico"))
00216
00217         while not self.passFlagUre or not self.passFlagCm:
00218             event, values = window.read()
00219
00220             if event == "Submit":
00221                 try:
00222                     self.__SetValues(values)
00223                 except Exception as e:
00224                     print(e)
00225                     RESTError(993)
00226                 finally:
00227                     pass
00228             elif event == sg.WIN_CLOSED or event == "Quit":
00229                 break
00230             else:
00231                 pass
00232
00233         window.close()
00234
00235     def __CreateFrame(self):
00236         """
00237         The __CreateFrame function creates the GUI layout for the Authentication Utility.
00238         It is called by __init__ and returns a list of lists that contains all the elements
00239         that will be displayed in the window.
00240
00241         Args:
00242             self: Access the class attributes and methods
00243
00244         Returns:
00245             A list of lists
00246
00247         Doc Author:
00248             Trelent
00249
00250         """
00251         sg.theme('Default1')
00252
00253         line00 = [sg.HSeparator()]
00254
00255         line0 = [sg.Image(ImageLoader("logo.png")),
00256                  sg.Push(),
00257                  sg.Text("Authentication Utility", font=("Helvetica", 12, "bold"),
00258                          justification="center"),

```

```

00257         sg.Push(),
00258         sg.Push()]
00259
00260     line1 = [sg.HSeparator()]
00261
00262     line2 = [sg.Push(),
00263             sg.Text("Utah Real Estate API Key: ", justification="center"),
00264             sg.Push()]
00265
00266     line3 = [sg.Push(),
00267             sg.Input(default_text="123", key="-ureAuth-", disabled=False,
00268                     size=(40, 1)),
00269             sg.Push()]
00270
00271     line4 = [sg.HSeparator()]
00272
00273     line5 = [sg.Push(),
00274             sg.Text("Construction Monitor HTTP BASIC Key: ", justification="center"),
00275             sg.Push()]
00276
00277     line6 = [sg.Push(),
00278             sg.Input(default_text="Basic 123", key="-cmAuth-", disabled=False,
00279                     size=(40, 1)),
00280             sg.Push()]
00281
00282     line7 = [sg.HSeparator()]
00283
00284     line8 = [sg.Push(),
00285             sg.Text(self.outcomeText, justification="center"),
00286             sg.Push()]
00287
00288     line9 = [sg.HSeparator()]
00289
00290     line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00291
00292     layout = [line00, line0, line1, line2, line3, line4, line5, line6, line7, line8, line9, line10]
00293
00294     return layout

```

4.4 BatchProcessing.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
00002 # http://www.apache.org/licenses/
00003
00004 import datetime
00005 import math
00006 from datetime import date
00007
00008 import pandas as pd
00009 import requests
00010
00011 from API_Calls.Functions.DataFunc.DataSupportFunctions import StringToList
00012
00013
00014 def BatchCalculator(TotalRecords, Argument_Dict):
00015     """
00016     The BatchCalculator function takes two arguments:
00017     1. TotalRecords - the total number of records in the database
00018     2. Argument_Dict - a dictionary containing all the arguments passed to this function by the user
00019
00020     Args:
00021     TotalRecords: Determine the number of batches that will be needed to complete the query
00022     Argument_Dict: Pass in the arguments that will be used to query the database
00023
00024     Returns:
00025     The total number of batches that will be made
00026
00027     Doc Author:
00028     Willem van der Schans, Trelent AI
00029     """
00030     try:
00031         document_limit = Argument_Dict["size"]
00032     except Exception as e:
00033         # Logging
00034         print(

```

```

00035         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")
00036         document_limit = 200
00037
00038         return int(math.ceil(float(TotalRecords) / float(document_limit)))
00039
00040
00041 class BatchProcessorConstructionMonitor:
00042
00043     def __init__(self, RestDomain, NumBatches, ParameterDict, HeaderDict, ColumnSelection, valueObject):
00044
00045         """
00046         The __init__ function is the constructor for a class. It is called when an object of that class
00047         is created, and it sets up the attributes of that object. In this case, we are setting up our
00048         object to have a dataframe attribute (which will be used to store all of our data), as well as
00049         attributes for each parameter in our ReST call.
00050
00051         Args:
00052             self: Represent the instance of the class
00053             RestDomain: Specify the domain of the rest api
00054             NumBatches: Determine how many batches of data to retrieve
00055             ParameterDict: Pass in the parameters that will be used to make the api call
00056             HeaderDict: Pass the header dictionary from the main function to this class
00057             ColumnSelection: Determine which columns to pull from the api
00058             valueObject: Pass in the value object that is used to determine what values are returned
00059
00060         Returns:
00061             An object of the class
00062
00063         Doc Author:
00064             Willem van der Schans, Trelent AI
00065         """
00066         self.dataframe = None
00067         self.__numBatches = NumBatches
00068         self.__parameterDict = ParameterDict
00069         self.__restDomain = RestDomain
00070         self.__headerDict = HeaderDict
00071         self.__columnSelection = ColumnSelection
00072         self.valueObject = valueObject
00073         self.__maxRequests = 10000
00074         self.__requestCount = math.ceil(self.__numBatches / (self.__maxRequests /
int(self.__parameterDict['size'])))
00075         self.__requestCalls = math.ceil(self.__maxRequests / int(self.__parameterDict['size']))
00076         self.__dateTracker = None
00077
00078     def FuncSelector(self):
00079         """
00080         The FuncSelector function is a function that takes the valueObject and passes it to the
ConstructionMonitorProcessor function.
00081         The ConstructionMonitorProcessor function then uses this valueObject to determine which of its
functions should be called.
00082
00083         Args:
00084             self: Represent the instance of the class
00085
00086         Returns:
00087             The result of the constructionmonitorprocessor function
00088
00089         Doc Author:
00090             Willem van der Schans, Trelent AI
00091         """
00092         self.ConstructionMonitorProcessor(self.valueObject)
00093
00094     def ConstructionMonitorProcessor(self, valueObject):
00095         """
00096         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
00097         process will be repeated until all the data has been collected from ConstructionMonitor.com's ReST
API, at which point __df will contain all
00098
00099         Args:
00100             self: Represent the instance of the object itself
00101             valueObject: Update the progress bar in the gui
00102
00103         Returns:
00104             A dataframe
00105
00106         Doc Author:
00107             Willem van der Schans, Trelent AI
00108         """
00109

```

```

00110     __df = None
00111     for callNum in range(0, self.__requestCount):
00112         self.__parameterDict["from"] = 0
00113
00114         if self.__requestCount > 1 and callNum != self.__requestCount - 1:
00115             __batchNum = self.__requestCalls
00116             if __df is None:
00117                 self.__dateTracker = str(date.today())
00118             else:
00119                 self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00120         elif self.__requestCount == 1:
00121             __batchNum = self.__numBatches
00122             self.__dateTracker = str(date.today())
00123         else:
00124             __batchNum = self.__numBatches / (self.__maxRequests / int(self.__parameterDict['size']))
00125     - (
00126         self.__requestCount - 1)
00127     self.__dateTracker = min(pd.to_datetime(__df['lastIndexedDate'])).strftime('%Y-%m-%d')
00128     self.__parameterDict['dateEnd'] = self.__dateTracker
00129
00130     for record in range(0, int(math.ceil(__batchNum))):
00131         if record != 0:
00132             self.__parameterDict["from"] = record * int(self.__parameterDict["size"])
00133
00134         response = requests.post(url=self.__restDomain,
00135                                 headers=self.__headerDict,
00136                                 json=self.__parameterDict)
00137
00138         counter = 0
00139         try:
00140             response = response.json()['hits']['hits']
00141         except KeyError as e:
00142             # Logging
00143             print(
00144                 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}")
00145             continue
00146
00147         valueObject.setValue(valueObject.getValue() + 1)
00148
00149         if record == 0 and callNum == 0:
00150             __df = pd.json_normalize(response[counter]["_source"])
00151             __df["id"] = response[counter]['_id']
00152             __df["county"] = response[counter]["_source"]['county']['county_name']
00153             counter += 1
00154
00155         for i in range(counter, len(response)):
00156             __tdf = pd.json_normalize(response[i]["_source"])
00157             __tdf["id"] = response[i]['_id']
00158             __tdf["county"] = response[i]["_source"]['county']['county_name']
00159             __df = pd.concat([__df, __tdf], ignore_index=True)
00160
00161         if self.__columnSelection is not None:
00162             __col_list = StringToList(self.__columnSelection)
00163             __col_list.append("id")
00164             __col_list.append("county")
00165         else:
00166             pass
00167
00168         self.dataframe = __df
00169         valueObject.setValue(-999)
00170
00171
00172 class BatchProcessorUtahRealEstate:
00173
00174     def __init__(self, RestDomain, NumBatches, ParameterString, HeaderDict, valueObject):
00175         """
00176         The __init__ function is the constructor for a class. It is called when an object of that class
00177         is instantiated, and it sets up the attributes of that object. In this case, we are setting up
00178         the dataframe attribute to be None (which will be set later), and we are also setting up some
00179         other attributes which will help us make our API calls.
00180
00181         Args:
00182             self: Represent the instance of the class
00183             RestDomain: Specify the domain of the rest api
00184             NumBatches: Determine how many batches of data to pull from the api
00185             ParameterString: Pass the parameters to the rest api
00186             HeaderDict: Pass in the header information for the api call
00187             valueObject: Create a dataframe from the json response

```

```

00188
00189     Returns:
00190         The instance of the class
00191
00192     Doc Author:
00193         Willem van der Schans, Trelent AI
00194     """
00195     self.dataframe = None
00196     self.__numBatches = NumBatches
00197     self.__parameterString = ParameterString
00198     self.__restDomain = RestDomain
00199     self.__headerDict = HeaderDict
00200     self.valueObject = valueObject
00201
00202     def FuncSelector(self):
00203         """
00204         The FuncSelector function is a function that takes the valueObject as an argument and then calls the
00205         appropriate function based on what was selected in the dropdown menu. The valueObject is passed to each of
00206         these functions so that they can access all of its attributes.
00207
00208     Args:
00209         self: Represent the instance of the class
00210
00211     Returns:
00212         The function that is selected by the user
00213
00214     Doc Author:
00215         Willem van der Schans, Trelent AI
00216     """
00217     self.BatchProcessingUtahRealestateCom(self.valueObject)
00218
00219     def BatchProcessingUtahRealestateCom(self, valueObject):
00220         """
00221         The BatchProcessingUtahRealestateCom function is a function that takes in the valueObject and uses it
00222         to update the progress bar. It also takes in self, which contains all the necessary information for
00223         this function to work properly. The BatchProcessingUtahRealestateCom function will then use requests to
00224         get data from UtahRealestate.com's ReST API and store it into a pandas DataFrame object called __df (which is
00225         local). This process will be repeated until all the data has been collected from UtahRealestate.com's ReST API,
00226         at which point __df will contain all
00227
00228     Args:
00229         self: Represent the instance of the class
00230         valueObject: Pass the value of a progress bar to the function
00231
00232     Returns:
00233         A dataframe of the scraped data
00234
00235     Doc Author:
00236         Willem van der Schans, Trelent AI
00237     """
00238     __df = pd.DataFrame()
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
00250 * 200}",
00251                                     headers=self.__headerDict)
00252
00253             response_temp = response.json()
00254             response_temp = pd.json_normalize(response_temp, record_path=['value'])
00255             __df = pd.concat([__df, response_temp], ignore_index=True)
00256
00257             valueObject.setValue(valueObject.getValue() + 1)
00258
00259     self.dataframe = __df
00260     valueObject.setValue(-999)

```


4.5 DataSupportFunctions.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002
00003
00004 def StringToList(string):
00005     """
00006     The StringToList function takes a string and converts it into a list.
00007     The function is used to convert the input from the user into a list of strings, which can then be
      iterated through.
00008
00009     Args:
00010         string: Split the string into a list
00011
00012     Returns:
00013         A list of strings
00014
00015     Doc Author:
00016         Willem van der Schans, Trelent AI
00017     """
00018     listOut = list(string.split(","))
00019     return listOut

```

4.6 FileSaver.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import os
00006 from pathlib import Path
00007
00008 import pandas as pd
00009
00010 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00011
00012
00013 class FileSaver:
00014
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(
      datetime.datetime.today().strftime('%m%d%Y'))
00036     self.data = outputDF
00037     self.dataAppending = None
00038     self.appendFlag = True
00039     self.fileName = f"{method}_{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.csv"
00040     self.uiFlag = True
00041
00042
00043     if method.lower() == "ure":
00044         self.primaryKey = "ListingKeyNumeric"
00045     elif method.lower() == "cm":
00046         self.primaryKey = "id"
00047     elif "realtor" in method.lower():

```

```

00048         self.primaryKey = None
00049         self.uiFlag = False
00050     elif method.lower() == "cfbp":
00051         self.primaryKey = None
00052         self.uiFlag = False
00053     else:
00054         raise ValueError("method input is invalid choice one of 4 options: URE, CM, Realtor, CFBP")
00055
00056     if AppendingPath is None:
00057         self.appendFlag = False
00058     else:
00059         self.dataAppending = pd.read_csv(AppendingPath)
00060
00061     if self.appendFlag:
00062         if self.primaryKey is not None:
00063             # Due to low_memory loading the columns are not typed properly,
00064             # since we are comparing this will be an issue since we need to do type comparisons,
00065             # so here we coerce the types of the primary keys to numeric.
00066             # If another primary key is ever chosen make sure to core to the right data type.
00067             self.dataAppending[self.primaryKey] = pd.to_numeric(self.dataAppending[self.primaryKey])
00068             self.data[self.primaryKey] = pd.to_numeric(self.data[self.primaryKey])
00069
00070             self.outputFrame = pd.concat([self.dataAppending,
self.data]).drop_duplicates(subset=[self.primaryKey],
00071                                     keep="last")
00072         else:
00073             self.outputFrame = pd.concat([self.dataAppending, self.data]).drop_duplicates(keep="last")
00074     else:
00075         self.outputFrame = self.data
00076
00077     if os.path.exists(self.docPath):
00078         self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
00079     else:
00080         os.mkdir(self.docPath)
00081         self.outputFrame.to_csv(self.docPath.joinpath(self.fileName), index=False)
00082
00083     if self.uiFlag:
00084         if self.appendFlag:
00085             PopupWrapped(text=f"File Appended and Saved to {self.docPath.joinpath(self.fileName)}",
00086                           windowType="savedLarge")
00087
00088             # Logging
00089             print(
00090                 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")
00091             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]}")
00092         else:
00093             PopupWrapped(text=f"File Saved to {self.docPath.joinpath(self.fileName)}",
windowType="savedLarge")
00094
00095             # Logging
00096             print(
00097                 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")
00098     else:
00099         pass
00100
00101     def getPath(self):
00102         """
00103         The getPath function returns the path to the file.
00104         It is a string, and it joins the docPath with the fileName.
00105
00106         Args:
00107             self: Represent the instance of the class
00108
00109         Returns:
00110             The path to the file
00111
00112         Doc Author:
00113             Willem van der Schans, Trelent AI
00114         """
00115         return str(self.docPath.joinpath(self.fileName))

```

4.7 Settings.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002
00003
00004 # Setting NameSpace for maintenance
00005
00006 class settings:
00007     # Version Checker
00008     # Update accordingly using semantic versioning: https://semver.org/
00009     settingVersion = "1.2.0"
00010     # Update in conjunction with settingDownloadSourceLink
00011     settingGithubApiUrl = "https://api.github.com/repos/Kydoimos97/GardnerApiUtility/releases/latest"
00012
00013     # PopUpWrapped
00014     # Singular Reference free to change
00015     settingGenerationToolLink = 'https://www.debugbear.com/basic-auth-header-generator'
00016     # Update in conjunction with settingDownloadSourceLink
00017     settingDownloadSourceLink = 'https://github.com/Kydoimos97/GardnerApiUtility/releases/latest'
00018
00019     # CFBP Source
00020     # This link downloads csv's immediately so minimal change is likely required in the source code
00021     settingCFBPLink = "https://ffiec.cfbp.gov/v2/data-browser-api/view/csv?"
00022
00023     # ConstructionMonitor Source
00024     # Check the REST call and data parser when updating this
00025     settingCMRestDomain = "https://api.constructionmonitor.com/v2/powersearch/"
00026
00027     # Realtor.Com Source
00028     # Updating This link likely requires a rewrite of the html parser
00029     settingRealtorLink = "https://www.realtor.com/research/data/"
00030
00031     # UtahRealEstate Source
00032     # API links are generated with hard references so updating this link requires a large code
      rewrite
00033     settingURERestDomain = "https://resoapi.utahrealestate.com/reso/odata/Property?"

```

4.8 versionChecker.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002 import requests
00003
00004 from API_Calls.Functions.DataFunc.Settings import settings
00005 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00006
00007
00008 def versionChecker():
00009     """
00010     The versionChecker function is used to check if the current version of the program is up-to-date.
00011     It does this by comparing the latest release on GitHub.
00012     If they are not equal, it will pop up a window telling you that there's an update available.
00013
00014     Args:
00015
00016     Returns:
00017         A popup window with the current version and latest version
00018
00019     Doc Author:
00020     Willem van der Schans, Trelent AI
00021     """
00022     # Todo Gitlab Update
00023     current_version = settings.settingVersion
00024     # Todo Gitlab Update
00025     response = requests.get(settings.settingGithubApiUrl)
00026     latest_version = response.json()['name']
00027     text_string = f"A different version is tagged as latest release \n \n" \
00028                 f"Running version: {current_version}\n" \
00029                 f"Latest version: {latest_version}"
00030     print(text_string)
00031
00032     if current_version != latest_version:
00033         PopupWrapped(text_string, windowType="versionWindow")

```

4.9 ErrorPopup.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002
00003
00004 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00005
00006
00007 def ErrorPopup(textString):
00008     """
00009     The ErrorPopup function is used to display a popup window with an error message.
00010     It takes one argument, textString, which is the string that will be displayed in the popup window.
00011     The function also opens up the log folder upon program exit.
00012
00013     Args:
00014         textString: Display the error message
00015
00016     Returns:
00017         Nothing, but it does print an error message to the console
00018
00019     Doc Author:
00020         Willem van der Schans, Trelent AI
00021     """
00022     PopupWrapped(
00023         f"ERROR @ {textString} \n"
00024         f"Log folder will be opened upon program exit",
00025         windowType="FatalErrorLarge")

```

4.10 ErrorPrint.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005
00006
00007 def RESTErrorPrint(response):
00008     """
00009     The RESTErrorPrint function is used to print the response from a ReST API call.
00010     If the response is an integer, it will be printed as-is. If it's not an integer,
00011     it will be converted to text and then printed.
00012
00013     Args:
00014         response: Print the response from a rest api call
00015
00016     Returns:
00017         The response text
00018
00019     Doc Author:
00020         Willem van der Schans, Trelent AI
00021     """
00022     if isinstance(response, int):
00023         print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Resource Response:
{response}")
00024     else:
00025         response_txt = response.text
00026         print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Resource Response:
{response_txt}")

```

4.11 Logger.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
      http://www.apache.org/licenses/
00002
00003
00004 import datetime
00005 import os
00006 import sys
00007 from pathlib import Path
00008

```

```

00009
00010 def logger():
00011     """
00012     The logger function creates a log file in the user's AppData directory.
00013     The function will create the directory if it does not exist.
00014     The function will also delete the oldest file when 100 logs have been saved to prevent bloat.
00015
00016     Args:
00017
00018     Returns:
00019         A file path to the log file that was created
00020
00021     Doc Author:
00022         Willem van der Schans, Trelent AI
00023     """
00024     dir_path = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs'))
00025     if os.path.exists(dir_path):
00026         pass
00027     else:
00028         if os.path.exists(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil'))):
00029             os.mkdir(dir_path)
00030         else:
00031             os.mkdir(Path(os.path.expandvars(r'%APPDATA%\GardnerUtil')))
00032             os.mkdir(dir_path)
00033
00034     filePath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Logs')).joinpath(
00035         f"{datetime.datetime.today().strftime('%m%d%Y_%H%M%S')}.log")
00036     sys.stdout = open(filePath, 'w')
00037     sys.stderr = sys.stdin = sys.stdout
00038
00039     def sorted_ls(path):
00040         """
00041         The sorted_ls function takes a path as an argument and returns the files in that directory sorted by
00042         modification time.
00043
00044         Args:
00045             path: Specify the directory to be sorted
00046
00047         Returns:
00048             A list of files in a directory sorted by modification time
00049
00050         Doc Author:
00051             Willem van der Schans, Trelent AI
00052         """
00053         mtime = lambda f: os.stat(os.path.join(path, f)).st_mtime
00054         return list(sorted(os.listdir(path), key=mtime))
00055
00056     del_list = sorted_ls(dir_path)[0:(len(sorted_ls(dir_path)) - 100)]
00057     for file in del_list:
00058         os.remove(dir_path.joinpath(file))
00059         print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Log file {file}
00060         deleted")

```

4.12 RESTError.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
00002     http://www.apache.org/licenses/
00003
00004 import datetime
00005
00006 from API_Calls.Functions.ErrorFunc.ErrorPopup import ErrorPopup
00007 from API_Calls.Functions.ErrorFunc.ErrorPrint import RESTErrorPrint
00008
00009
00010 def RESTError(response):
00011     """
00012     The RESTError function is a function that checks the status codes.
00013     If it is 200, then everything went well and nothing happens. If it isn't 200, then an error message will
00014     be printed to
00015     the console with information about what happened (i.e., if there was an authentication error or if the
00016     resource wasn't found).
00017     The function also raises an exception and opens an error popup for easy debugging.
00018
00019     Args:
00020         response: Print out the response from the server
00021
00022     Returns:
00023         None
00024
00025     Doc Author:
00026         Willem van der Schans, Trelent AI
00027     """

```

```

00020 Returns:
00021     A text string
00022
00023 Doc Author:
00024     Trelent
00025 """
00026     if isinstance(response, int):
00027         status_code = response
00028     else:
00029         status_code = response.status_code
00030
00031     if status_code == 200:
00032         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | Api Request completed successfully"
00033         print(textString)
00034         pass
00035     elif status_code == 301:
00036         RESTErrorPrint(response)
00037         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"
00038         ErrorPopup(textString)
00039         raise ValueError(textString)
00040     elif status_code == 400:
00041         RESTErrorPrint(response)
00042         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | Bad Request; check input arguments"
00043         ErrorPopup(textString)
00044         raise ValueError(textString)
00045     elif status_code == 401:
00046         RESTErrorPrint(response)
00047         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | Authentication Error: No keys found"
00048         ErrorPopup(textString)
00049         raise PermissionError(textString)
00050     elif status_code == 402:
00051         RESTErrorPrint(response)
00052         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"
00053         ErrorPopup(textString)
00054         raise PermissionError(textString)
00055     elif status_code == 403:
00056         RESTErrorPrint(response)
00057         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"
00058         ErrorPopup(textString)
00059         raise PermissionError(textString)
00060     elif status_code == 404:
00061         RESTErrorPrint(response)
00062         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"
00063         ErrorPopup(textString)
00064         raise NameError(textString)
00065     elif status_code == 405:
00066         RESTErrorPrint(response)
00067         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"
00068         ErrorPopup(textString)
00069         raise ValueError(textString)
00070     elif status_code == 408:
00071         RESTErrorPrint(response)
00072         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | Requests timeout by server"
00073         ErrorPopup(textString)
00074         raise TimeoutError(textString)
00075     elif status_code == 503:
00076         RESTErrorPrint(response)
00077         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"
00078         ErrorPopup(textString)
00079         raise SystemError(textString)
00080     elif status_code == 701:
00081         RESTErrorPrint(response)
00082         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)"
00083         ErrorPopup(textString)
00084         raise TypeError(textString)
00085     elif status_code == 801:
00086         RESTErrorPrint(response)
00087         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"

```

```

00088         ErrorPopup(textString)
00089         raise ValueError(textString)
00090     elif status_code == 790:
00091         RESTErrorPrint(response)
00092         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | Requests timeout within requests"
00093         ErrorPopup(textString)
00094         raise TimeoutError(textString)
00095     elif status_code == 791:
00096         RESTErrorPrint(response)
00097         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | Too many redirects, Bad url"
00098         ErrorPopup(textString)
00099         raise ValueError(textString)
00100     elif status_code == 990:
00101         RESTErrorPrint(response)
00102         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | No password input"
00103         ErrorPopup(textString)
00104         raise ValueError(textString)
00105     elif status_code == 991:
00106         RESTErrorPrint(response)
00107         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | No username input"
00108         ErrorPopup(textString)
00109         raise ValueError(textString)
00110     elif status_code == 992:
00111         RESTErrorPrint(response)
00112         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)"
00113         ErrorPopup(textString)
00114         raise ValueError(textString)
00115     elif status_code == 993:
00116         RESTErrorPrint(response)
00117         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"
00118         ErrorPopup(textString)
00119         print(ValueError(textString))
00120     elif status_code == 994:
00121         RESTErrorPrint(response)
00122         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | Submission Error: server returned no documents"
00123         ErrorPopup(textString)
00124         raise ValueError(textString)
00125     elif status_code == 1000:
00126         RESTErrorPrint(response)
00127         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | Catastrophic Error"
00128         ErrorPopup(textString)
00129         raise SystemError(textString)
00130     elif status_code == 1001:
00131         RESTErrorPrint(response)
00132         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | Main Function Error Break"
00133         raise SystemError(textString)
00134     elif status_code == 1100:
00135         RESTErrorPrint(response)
00136         textString = f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Status Code =
{status_code} | User has cancelled the program execution"
00137         raise KeyboardInterrupt(textString)
00138     elif status_code == 1101:
00139         RESTErrorPrint(response)
00140         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"
00141         print(textString)
00142     else:
00143         RESTErrorPrint(response)
00144         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

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
http://www.apache.org/licenses/
00002
00003 import PySimpleGUI as sg
00004

```

```

00005 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00006
00007
00008 def BatchInputGui(batches, documentCount=None):
00009     """
00010     The BatchInputGui function is a simple GUI that displays the number of batches and pages
00011     that will be requested. It also gives the user an option to cancel or continue with their request.
00012
00013
00014     Args:
00015         batches: Determine how many batches will be run
00016         documentCount: Determine how many documents will be retrieved
00017
00018     Returns:
00019         The event, which is the button that was pressed
00020
00021     Doc Author:
00022         Willem van der Schans, Trelent AI
00023     """
00024     event = None
00025     if documentCount is None:
00026         __text1 = f"This request will run {batches}"
00027     else:
00028         __text1 = f"This request will run {batches} batches and will retrieve {documentCount} rows"
00029
00030     __text2 = "Press Continue to start request"
00031
00032     __Line1 = [sg.Push(),
00033               sg.Text(__text1, justification="center"),
00034               sg.Push()]
00035
00036     __Line2 = [sg.Push(),
00037               sg.Text(__text2, justification="center"),
00038               sg.Push()]
00039
00040     __Line3 = [sg.Push(),
00041               sg.Ok("Continue"),
00042               sg.Cancel(),
00043               sg.Push()]
00044
00045     window = sg.Window("Popup", [__Line1, __Line2, __Line3],
00046                       modal=True,
00047                       keep_on_top=True,
00048                       disable_close=True,
00049                       icon=ImageLoader("taskbar_icon.ico"))
00050
00051     while True:
00052         event, values = window.read()
00053         if event == "Continue":
00054             break
00055         elif event == sg.WIN_CLOSED or event == "Cancel":
00056             break
00057
00058     window.close()
00059
00060     return event
00061
00062
00063 def confirmDialog():
00064     """
00065     The confirmDialog function is a simple confirmation dialog that asks the user if they want to continue
00066     with the request.
00067     The function takes no arguments and returns the button event to allow for process confirmation.
00068
00069
00070     Args:
00071
00072
00073     Returns:
00074         The event that was triggered,
00075
00076
00077     Doc Author:
00078         Willem van der Schans, Trelent AI
00079     """
00080     event = None
00081     __text1 = f"This request can take multiple minutes to complete"
00082     __text2 = "Press Continue to start the request"
00083
00084     __Line1 = [sg.Push(),
00085               sg.Text(__text1, justification="center"),
00086               sg.Push()]
00087
00088     __Line2 = [sg.Push(),

```



```

00085         sg.Text(__text2, justification="center"),
00086         sg.Push()]
00087
00088     __Line3 = [sg.Push(),
00089               sg.Ok("Continue"),
00090               sg.Cancel(),
00091               sg.Push()]]
00092
00093     window = sg.Window("Popup", [__Line1, __Line2, __Line3],
00094                          modal=True,
00095                          keep_on_top=True,
00096                          disable_close=True,
00097                          icon=ImageLoader("taskbar_icon.ico"))
00098
00099     while True:
00100         event, values = window.read()
00101         if event == "Continue":
00102             break
00103         elif event == sg.WIN_CLOSED or event == "Cancel":
00104             break
00105
00106     window.close()
00107
00108     return event

```

4.14 BatchProgressGUI.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
00002 # http://www.apache.org/licenses/
00003
00003 import datetime
00004 import threading
00005 import time
00006
00007 import PySimpleGUI as sg
00008
00009 from API_Calls.Functions.DataFunc.BatchProcessing import BatchProcessorConstructionMonitor,
00010 BatchProcessorUtahRealEstate
00011 from API_Calls.Functions.Gui.DataTransfer import DataTransfer
00012 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00013 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00014
00014 counter = 1
00015
00016
00017 class BatchProgressGUI:
00018
00019     def __init__(self, BatchesNum, RestDomain, ParameterDict, HeaderDict, Type, ColumnSelection=None):
00020
00021         """
00022         The __init__ function is the first function that gets called when an object of this class is created.
00023         It initializes all the variables and sets up a layout for the GUI. It also creates a window to display
00024         the dataframe in.
00025
00026         Args:
00027             self: Represent the instance of the class
00028             BatchesNum: Determine the number of batches that will be created
00029             RestDomain: Specify the domain of the rest api
00030             ParameterDict: Pass the parameters of the request to the class
00031             HeaderDict: Store the headers of the dataframe
00032             Type: Determine the type of dataframe that is being created
00033             ColumnSelection: Select the columns to be displayed in the gui
00034
00035         Returns:
00036             Nothing
00037
00038         Doc Author:
00039             Willem van der Schans, Trelent AI
00040
00041         """
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
00053     def BatchGuiShow(self):
00054         """
00055         The BatchGuiShow function is called by the BatchGui function. It creates a progress bar layout and
00056         then calls the createGui function to create a GUI for batch processing.
00057
00058         Args:
00059             self: Represent the instance of the class
00060
00061         Returns:
00062             The __type of the batchgui class
00063
00064         Doc Author:
00065             Willem van der Schans, Trelent AI
00066         """
00067         self.CreateProgressLayout()
00068         self.createGui(self.__type)
00069
00070     def CreateProgressLayout(self):
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",
00090         key="--progress_text--"),
00091         sg.Push()]
00092
00093         __Line2 = [sg.Push(), sg.Text(font=("Helvetica", 10), justification="center", key="--timer--"),
00094         sg.Text(font=("Helvetica", 10), justification="center", key="--time_est--"), sg.Push()]
00095
00096         __Line3 = [
00097             sg.ProgressBar(max_value=self.__batches, bar_color=("#920303", "#C9c8c8"), orientation='h',
00098             size=(30, 20),
00099             key='--progress_bar--')]
00100
00101         layout = [__Line1, __Line2, __Line3]
00102
00103         self.__layout = layout
00104
00105     def createGui(self, Sourcetype):
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
00113         these threads.
00114
00115         Args:
00116             self: Access the object itself
00117             Sourcetype: Determine which batch processor to use
00118
00119         Returns:
00120             The dataframe
00121
00122         Doc Author:
00123             Willem van der Schans, Trelent AI
00124         """
00125         self.__window = sg.Window('Progress', self.__layout, finalize=True,
00126         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
00144     elif Sourcetype == "utah_real_estate":
00145         processorObject = BatchProcessorUtahRealEstate(RestDomain=self.__restDomain,
00146                                                         NumBatches=self.__batches,
00147                                                         ParameterString=self.__parameterDict,
00148                                                         HeaderDict=self.__headerDict,
00149                                                         valueObject=valueObj)
00150
00151     threading.Thread(target=self.TimeUpdater,
00152                     args=(start_time,),
00153                     daemon=True).start()
00154     print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | TimeUpdater Thread
00155     Successfully Started")
00156
00157     batchFuncThread = threading.Thread(target=processorObject.FuncSelector,
00158                                       daemon=False)
00159     batchFuncThread.start()
00160     print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | BatchFunc Thread
00161     Successfully Started")
00162     threading.Thread(target=self.ValueChecker,
00163                     args=(valueObj,),
00164                     daemon=False).start()
00165     print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | ValueChecker Thread
00166     Successfully Started")
00167
00168     while True:
00169
00170         self.ProgressUpdater(valueObj)
00171
00172         if valueObj.getValue() == -999:
00173             break
00174
00175         window, event, values = sg.read_all_windows()
00176         if event.startswith('update'):
00177             __key_to_update = event[len('update'):]
00178             window[__key_to_update].update(values[event])
00179             window.refresh()
00180             pass
00181
00182         if event == sg.WIN_CLOSED or event == "Cancel" or event == "Exit":
00183             break
00184
00185         time.sleep(0.1)
00186
00187     self.dataframe = processorObject.dataframe
00188     self.__window.close()
00189
00190     PopupWrapped(text="Api Request Completed", windowType="notice")
00191
00192     def ProgressUpdater(self, valueObj):
00193         """
00194         The ProgressUpdater function is a callback function that updates the progress bar and text
00195         in the GUI. It takes in one argument, which is an object containing information about the
00196         current batch number. The ProgressUpdater function then checks if this value has changed from
00197         the last time it was called (i.e., if we are on a new batch). If so, it updates both the progress
00198         bar and text with this new information.
00199
00200         Args:
00201             self: Make the progressupdater function an instance method
00202             valueObj: Get the current value of the batch counter
00203
00204         Returns:
00205             The value of the batch counter
00206

```

```

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
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
00220     complete the current batch. It does this by reading the start_time variable passed in, getting the current
00221     time, calculating how much time has passed since start_time was set and then updating a timer string with
00222     that value. It then calculates an estimation of how long it will take to finish all batches based on how many
00223     batches have been completed so far.
00224
00225     Args:
00226     self: Make the function a method of the class
00227     start_time: Get the time when the function is called
00228
00229     Returns:
00230     A string that is updated every 0
00231
00232     Doc Author:
00233     Willem van der Schans, Trelent AI
00234     """
00235     while True:
00236         if self.__batch_counter < self.__batches:
00237             __current_time = datetime.datetime.now().replace(microsecond=0)
00238
00239             __passed_time = __current_time - start_time
00240
00241             __timer_string = f"Time Elapsed {__passed_time}"
00242
00243             try:
00244                 self.__window.write_event_value('update--timer--', __timer_string)
00245             except AttributeError as e:
00246                 print(
00247                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} |
00248                     BatchProgressGUI.py | Error = {e} | Timer string attribute error, this is okay if the display looks good,
00249                     this exception omits fatal crashes due to an aesthetic error")
00250                 break
00251
00252             __passed_time = __passed_time.total_seconds()
00253
00254             try:
00255                 __time_est = datetime.timedelta(
00256                     seconds=(__passed_time * (self.__batches / self.__batch_counter) -
00257                         __passed_time)).seconds
00258             except:
00259                 __time_est = datetime.timedelta(
00260                     seconds=(__passed_time * self.__batches - __passed_time)).seconds
00261
00262             __time_est = time.strftime('%H:%M:%S', time.gmtime(__time_est))
00263
00264             __end_string = f"Est time needed {__time_est}"
00265             self.__window.write_event_value('update--time_est--', __end_string)
00266         else:
00267             __end_string = f"Est time needed 00:00:00"
00268             self.__window.write_event_value('update--time_est--', __end_string)
00269             time.sleep(0.25)
00270
00271 def ValueChecker(self, ObjectVal):
00272
00273     """
00274     The ValueChecker function is a thread that checks the value of an object.
00275     It will check if the value has changed, and if it has, it will return True.
00276     If not, then it returns False.
00277
00278     Args:
00279     self: Represent the instance of the class
00280     ObjectVal: Get the value of the object

```

```

00277
00278     Returns:
00279         True if the value of the object has changed, and false if it hasn't
00280
00281     Doc Author:
00282         Willem van der Schans, Trelent AI
00283     """
00284     while True:
00285         time.sleep(0.3)
00286         if self.__batch_counter != ObjectVal.getValue():
00287             self.__batch_counter = ObjectVal.getValue()
00288             return True
00289         else:
00290             return False

```

4.15 DataTransfer.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
00002     http://www.apache.org/licenses/
00003
00004 class DataTransfer:
00005
00006     def __init__(self):
00007         """
00008         The __init__ function is called when the class is instantiated.
00009         It sets the initial value of self.__value to 0.
00010
00011         Args:
00012             self: Represent the instance of the class
00013
00014         Returns:
00015             Nothing
00016
00017         Doc Author:
00018             Willem van der Schans, Trelent AI
00019         """
00020         self.__value = 0
00021
00022     def setValue(self, value):
00023         """
00024         The setValue function sets the value of the object.
00025
00026         Args:
00027             self: Represent the instance of the class
00028             value: Set the value of the instance variable __value
00029
00030         Returns:
00031             The value that was passed to it
00032
00033         Doc Author:
00034             Willem van der Schans, Trelent AI
00035         """
00036         self.__value = value
00037
00038
00039     def getValue(self):
00040         """
00041         The getValue function returns the value of the private variable __value.
00042         This is a getter function that allows access to this private variable.
00043
00044         Args:
00045             self: Represent the instance of the class
00046
00047         Returns:
00048             The value of the instance variable
00049
00050         Doc Author:
00051             Willem van der Schans, Trelent AI
00052         """
00053         return self.__value
00054
00055     def whileValue(self):
00056         """
00057         The whileValue function is a function that will run the getValue function until it is told to stop.
00058         This allows for the program to constantly be checking for new values from the sensor.
00059

```

```

00060     Args:
00061         self: Refer to the current instance of the class
00062
00063     Returns:
00064         The value of the input
00065
00066     Doc Author:
00067         Willem van der Schans, Trelent AI
00068     """
00069     while True:
00070         self.getValue()

```

4.16 ImageLoader.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
00002 # http://www.apache.org/licenses/
00003
00004 import base64
00005 import os
00006 from io import BytesIO
00007 from os.path import join, normpath
00008
00009 from PIL import Image
00010
00011
00012 def ImageLoader(file):
00013     """
00014     The ImageLoader function takes in a file name and returns the image as a base64 encoded string.
00015     This is used to send images to the API for processing.
00016
00017     Args:
00018         file: Specify the image file to be loaded
00019
00020     Returns:
00021         A base64 encoded image string
00022
00023     Doc Author:
00024         Willem van der Schans, Trelent AI
00025     """
00026     try:
00027         __path = normpath(join(str(os.getcwd()).split("API_Calls", 1)[0]), "API_Calls"))
00028         __path = normpath(join(__path, "External Files"))
00029         __path = normpath(join(__path, "Images"))
00030         __path = join(__path, file).replace("\\", "/")
00031
00032         image = Image.open(__path)
00033
00034         __buff = BytesIO()
00035
00036         image.save(__buff, format="png")
00037
00038         img_str = base64.b64encode(__buff.getvalue())
00039
00040         return img_str
00041     except Exception as e:
00042         # We cannot log this error like other errors due to circular imports
00043         raise e

```

4.17 PopupWrapped.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
00002 # http://www.apache.org/licenses/
00003
00004 import datetime
00005 import os
00006 import threading
00007 import time
00008 import webbrowser
00009 from pathlib import Path
00010
00011 import PySimpleGUI as sg
00012
00013 from API_Calls.Functions.DataFunc.Settings import settings

```

```

00012 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00013
00014
00015 class PopupWrapped():
00016
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
00058     def __createLayout(self):
00059         """
00060         The __createLayout function is used to create the layout of the window.
00061         The function takes class variables and returns a window layout.
00062         It uses a series of if statements to determine what type of window it is, then creates a layout based
on that information.
00063         Args:
00064             self: Refer to the current instance of a class
00065         Returns:
00066             A list of lists
00067         Doc Author:
00068             Willem van der Schans, Trelent AI
00069         """
00070         sg.theme('Default1')
00071         __Line1 = None
00072         __Line2 = None
00073
00074         if self.__type == "notice":
00075             __Line1 = [sg.Push(),
00076                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
00077                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00078             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00079         elif self.__type == "noticeLarge":
00080             __Line1 = [sg.Push(),
00081                       sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
00082                       sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00083             __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00084         elif self.__type == "savedLarge":
00085             if self.__errorFlag:
00086                 __Line1 = [sg.Push(),
00087                           sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
00088                           sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00089                 __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00090             else:

```

```

00091         __Line1 = [sg.Push(),
00092                     sg.Text(u'\u2713', font=("Helvetica", 20, "bold"), justification="center"),
00093                     sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00094         __Line2 = [sg.Push(), sg.Button("Open File", size=(10, 1)), sg.Ok(focus=True, size=(10,
1)), sg.Push()]
00095     elif self.__type == "errorLarge":
00096         __Line1 = [sg.Push(),
00097                     sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00098                     sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00099         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00100     elif self.__type == "FatalErrorLarge":
00101         __Line1 = [sg.Push(),
00102                     sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00103                     sg.Text(self.__text, justification="left", key="-textField-"), sg.Push()]
00104         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00105     elif self.__type == "error":
00106         __Line1 = [sg.Push(),
00107                     sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00108                     sg.Text(f"{self.__text}: {self.__error}", justification="center",
key="-textField-"),
00109                     sg.Push()]
00110         __Line2 = [sg.Push(), sg.Ok(focus=True, size=(10, 1)), sg.Push()]
00111     elif self.__type == "AuthError":
00112         __Line1 = [sg.Push(),
00113                     sg.Text(u'\u274C', font=("Helvetica", 20, "bold"), justification="center"),
00114                     sg.Text(f"{self.__text}", justification="center", key="-textField-"),
00115                     sg.Push()]
00116         __Line2 = [sg.Push(), sg.Button(button_text="Open Generation Tool [Web Browser]",
00117                                         sg.Ok(button_text="Return", focus=True, size=(10, 1)), sg.Push())]
00118     elif self.__type == "versionWindow":
00119         __Line1 = [sg.Push(),
00120                     sg.Text(f"{self.__text}", justification="left", key="-textField-"),
00121                     sg.Push()]
00122         __Line2 = [sg.Push(), sg.Button(button_text="Download"),
00123                     sg.Ok(button_text="Continue", focus=True, size=(10, 1)), sg.Push()]
00124     elif self.__type == "progress":
00125         __Line1 = [sg.Push(),
00126                     sg.Text(self.__text, justification="center", key="-textField-"), sg.Push()]
00127
00128     if self.__type == "progress":
00129         self.__layout = [__Line1, ]
00130     else:
00131         self.__layout = [__Line1, __Line2]
00132
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,
00179                                     finalize=True,
00180                                     modal=True,
00181                                     keep_on_top=True,
00182                                     disable_close=False,
00183                                     icon=ImageLoader("taskbar_icon.ico"))
00184     elif self.__type == "versionWindow":
00185         self.__windowObj = sg.Window(title="Update Notice", layout=self.__layout, finalize=True,
00186                                     modal=True,
00187                                     keep_on_top=True,
00188                                     disable_close=False,
00189                                     icon=ImageLoader("taskbar_icon.ico"))
00190     else:
00191         self.__windowObj = sg.Window(title=self.__type.capitalize(), layout=self.__layout,
00192                                     finalize=True,
00193                                     modal=True,
00194                                     keep_on_top=True,
00195                                     disable_close=False,
00196                                     icon=ImageLoader("taskbar_icon.ico"),
00197                                     size=(290, 80))
00198
00199     if self.__type != "progress" or self.__type.startswith("perm"):
00200         timer = 0
00201         while timer < 100:
00202             event, values = self.__windowObj.read()
00203             if event == "Ok" or event == sg.WIN_CLOSED or event == "Return" or event == "Continue":
00204                 break
00205             elif event == "Open Generation Tool [Web Browser]":
00206                 webbrowser.open(settings.settingGenerationToolLink, new=2, autoraise=True)
00207                 pass
00208             elif event == "Open File":
00209                 threadFile = threading.Thread(target=self.openFile,
00210                                             daemon=False)
00211                 threadFile.start()
00212                 time.sleep(3)
00213                 break
00214             elif event == "Download":
00215                 # Todo Gitlab Update
00216                 webbrowser.open(settings.settingDownloadSourceLink, new=2,
00217                               autoraise=True)
00218                 pass
00219             time.sleep(0.1)
00220
00221         if self.__type == "FatalErrorLarge":
00222             try:
00223                 os.system(
00224                     f"start
00225 {Path(os.path.expandvars(r'%APPDATA%')).joinpath('GardnerUtil').joinpath('Logs')}")
00226             except Exception as e:
00227                 print(
00228                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
00229 PopupWrapped.py | Error = {e} | Log Folder not found please search manually for
00230 %APPDATA%\Roaming\GardnerUtil\Logs\n")
00231
00232         self.__windowObj.close()
00233
00234     def stopWindow(self):
00235         """
00236         The stopWindow function is used to close the window object that was created in the startWindow
00237         function.
00238         This is done by calling the close() method on self.__windowObj, which will cause it to be destroyed.
00239         Args:
00240             self: Represent the instance of the class
00241         Returns:
00242             The window object
00243         Doc Author:
00244             Willem van der Schans, Trelent AI

```

```

00239     """
00240     self.__windowObj.close()
00241
00242     def textUpdate(self, sleep=0.5):
00243         """
00244         The textUpdate function is a function that updates the text in the text field.
00245         It does this by adding dots to the end of it, and then removing them. This creates
00246         a loading effect for when something is being processed.
00247         Args:
00248             self: Refer to the object itself
00249             sleep: Control the speed of the text update
00250         Returns:
00251             A string that is the current text of the text field
00252         Doc Author:
00253             Willem van der Schans, Trelent AI
00254         """
00255         self.__counter += 1
00256         if self.__counter == 4:
00257             self.__counter = 1
00258         newString = ""
00259         if self.__type == "notice":
00260             pass
00261         elif self.__type == "error":
00262             pass
00263         elif self.__type == "progress":
00264             newString = f"{self.__text}{'.' * self.__counter}"
00265         self.__windowObj.write_event_value('update-textField-', newString)
00266
00267         time.sleep(sleep)
00268
00269     def windowPush(self):
00270         """
00271         The windowPush function is used to update the values of a window object.
00272         The function takes in an event and values from the window object, then checks if the event starts
00273         with 'update'.
00274         If it does, it will take everything after 'update' as a key for updating that specific value.
00275         It will then update that value using its key and refresh the window.
00276         Args:
00277             self: Reference the object that is calling the function
00278         Returns:
00279             A tuple containing the event and values
00280         Doc Author:
00281             Willem van der Schans, Trelent AI
00282         """
00283         event, values = self.__windowObj.read()
00284
00285         if event.startswith('update'):
00286             __key_to_update = event[len('update'):]
00287             self.__windowObj[__key_to_update].update(values[event])
00288             self.__windowObj.refresh()
00289
00290     def openFile(self):
00291         """
00292         The openFile function opens the file that is associated with the
00293         document object. It does this by calling os.system and passing it
00294         self.__docpath as an argument.
00295
00296         Args:
00297             self: Represent the instance of the object itself
00298
00299         Returns:
00300             The filepath of the document
00301
00302         Doc Author:
00303             Willem van der Schans, Trelent AI
00304         """
00305         os.system(self.__docpath)

```

4.18 Initializer.py

```

00001 # This software is licensed under Apache License, Version 2.0, January 2004 as found on
00002 # http://www.apache.org/licenses/
00003
00004 import datetime
00005 import os

```

```

00006 from pathlib import Path
00007
00008 import PySimpleGUI as sg
00009
00010 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00011 from API_Calls.Functions.DataFunc.versionChecker import versionChecker
00012 from API_Calls.Functions.ErrorFunc.Logger import logger
00013 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00014 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00015 from API_Calls.Sources.CFBP.Core import CFBP
00016 from API_Calls.Sources.ConstructionMonitor.Core import ConstructionMonitorInit, \
00017     ConstructionMonitorMain
00018 from API_Calls.Sources.Realtor.Core import realtorCom
00019 from API_Calls.Sources.UtahRealEstate.Core import UtahRealEstateMain, UtahRealEstateInit
00020
00021
00022 class initializer:
00023
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
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
00056         elements:
00057         1) The type of element to be displayed (e.g., "Text", "InputText", etc.)
00058         2) A dictionary containing any additional parameters for that element (e.g., size, default value,
00059         etc.)
00060         3) An optional key name for the element (used in event handling). If no key name is provided then
00061         one will be generated automatically by PySimpleGUIQt based on its position in the layout list
00062
00063         Args:
00064             self: Represent the instance of the class
00065             layout: Pass the layout of the window to be created
00066             text: Set the title of the window
00067
00068         Returns:
00069             A window object
00070
00071         Doc Author:
00072             Willem van der Schans, Trelent AI
00073         """
00074         # Todo Gitlab Update
00075         versionChecker()
00076
00077         window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00078                             finalize=True,
00079                             icon=ImageLoader("taskbar_icon.ico"))
00080
00081         while True:
00082             event, values = window.read()
00083
00084             if event == "Construction Monitor":
00085                 print(
00086                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |

```

```

-----Initiating Construction Monitor API Call-----")
00084         ConstructionMonitorMain(ConstructionMonitorInit())
00085         print(
00086             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing Construction Monitor API Call-----\n")
00087         elif event == "Utah Real Estate":
00088             print(
00089                 f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Initiating Utah Real Estate API Call-----")
00090             UtahRealEstateMain(UtahRealEstateInit())
00091             print(
00092                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing Utah Real Estate API Call-----\n")
00093             elif event == "Realtor.Com":
00094                 print(
00095                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Initiating Realtor.com API Call-----")
00096                 realtorCom()
00097                 print(
00098                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing Realtor.com API Call-----\n")
00099             elif event == "CFPB Mortgage":
00100                 print(
00101                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Initiating ffiec.cfpb API Call-----")
00102                 CFPB()
00103                 print(
00104                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing ffiec.cfpb API Call-----\n")
00105             elif event == "Authorization Utility":
00106                 print(
00107                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Initiating Authorization Utility-----")
00108                 AuthUtil()
00109                 print(
00110                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Closing Authorization Utility-----\n")
00111             elif event == "Open Data Folder":
00112                 print(
00113                     f"\n{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
-----Data Folder Opened-----")
00114                 try:
00115                     os.system(f"start
(Path(os.path.expanduser('~\Documents')).joinpath('GardnerUtilData'))")
00116                 except:
00117                     try:
00118                         os.system(f"start {Path(os.path.expanduser('~\Documents'))}")
00119                     except Exception as e:
00120                         print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
Initializer.py | Error = {e} | Documents folder not found")
00121                         PopupWrapped(
00122                             text="Documents folder not found. Please create a Windows recognized documents
folder",
00123                             windowType="errorLarge")
00124
00125             elif event in ('Exit', None):
00126                 try:
00127                     break
00128                 except Exception as e:
00129                     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.")
00130                     break
00131             elif event == sg.WIN_CLOSED or event == "Quit":
00132                 break
00133
00134             window.close()
00135
00136         def __CreateFrame(self):
00137
00138             """
00139             The __CreateFrame function is a helper function that creates the layout for the main window.
00140             It returns a list of lists, which is then passed to sg.Window() as its layout parameter.
00141
00142             Args:
00143                 self: Represent the instance of the class
00144
00145             Returns:
00146                 A list of lists, which is then passed to the sg
00147
00148             Doc Author:
00149                 Willem van der Schans, Trelent AI

```

```

00150     """
00151         sg.theme('Default1')
00152
00153         line0 = [sg.HSeparator()]
00154
00155         line1 = [sg.Image(ImageLoader("logo.png")),
00156                 sg.Push(),
00157                 sg.Text("Gardner Data Utility", font=("Helvetica", 12, "bold"), justification="center"),
00158                 sg.Push(),
00159                 sg.Push()]
00160
00161         line3 = [sg.HSeparator()]
00162
00163         line4 = [sg.Push(),
00164                 sg.Text("Api Sources", font=("Helvetica", 10, "bold"), justification="center"),
00165                 sg.Push()]
00166
00167         line5 = [[sg.Push(), sg.Button("Construction Monitor", size=(20, None)), sg.Push(),
00168                 sg.Button("Utah Real Estate", size=(20, None)), sg.Push()]]
00169
00170         line6 = [[sg.Push(), sg.Button("Realtor.Com", size=(20, None)), sg.Push(),
00171                 sg.Button("CFPB Mortgage", size=(20, None)),
00172                 sg.Push()]]
00173
00174         line8 = [sg.HSeparator()]
00175
00176         line9 = [sg.Push(),
00177                 sg.Text("Utilities", font=("Helvetica", 10, "bold"), justification="center"),
00178                 sg.Push()]
00179
00180         line10 = [[sg.Push(), sg.Button("Authorization Utility", size=(20, None)),
00181                 sg.Button("Open Data Folder", size=(20, None)), sg.Push()]]
00182
00183         line11 = [sg.HSeparator()]
00184
00185         layout = [line0, line1, line3, line4, line5, line6, line8, line9, line10, line11]
00186
00187         return layout

```

4.19 CFBP/Core.py

```

00001 import datetime
00002 import threading
00003 import time
00004
00005 import pandas as pd
00006 import requests
00007
00008 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00009 from API_Calls.Functions.DataFunc.Settings import settings
00010 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00011 from API_Calls.Functions.Gui.BatchGui import confirmDialog
00012 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00013
00014
00015 class CFBP:
00016
00017     def __init__(self, state_arg=None, year_arg=None):
00018         """
00019         The __init__ function is called when the class is instantiated.
00020         Its job is to initialize the object with some default values, and do any other setup that might be
00021         necessary.
00022         The __init__ function can take arguments, but it doesn't have to.
00023
00024         Args:
00025             self: Represent the instance of the class
00026             state_arg: Set the state_arg attribute of the class
00027             year_arg: Set the year of data to be retrieved
00028
00029         Returns:
00030             A popupwrapped object
00031
00032         Doc Author:
00033             Willem van der Schans, Trelent AI
00034
00035         """
00036         self.state_arg = state_arg
00037         self.year_arg = year_arg

```

```

00036         self.uiString = None
00037         self.link = None
00038
00039         eventReturn = confirmDialog()
00040         if eventReturn == "Continue":
00041             startTime = datetime.datetime.now().replace(microsecond=0)
00042             self.__showUi()
00043             print(
00044                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | API Link =
{self.link}")
00045             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.cfbp.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
00058     def __showUi(self):
00059         """
00060         The __showUi function is a function that creates a progress bar window.
00061         The __showUi function takes class variables and returns a windowobj.
00062
00063         Args:
00064             self: Represent the instance of the class
00065
00066         Returns:
00067             The uiobj variable
00068
00069         Doc Author:
00070             Willem van der Schans, Trelent AI
00071         """
00072         uiObj = PopupWrapped(text="Cenus Request running", windowType="progress", error=None)
00073
00074         threadGui = threading.Thread(target=self.__dataGetter,
00075                                     daemon=False)
00076         threadGui.start()
00077
00078         while threadGui.is_alive():
00079             uiObj.textUpdate()
00080             uiObj.windowPush()
00081         else:
00082             uiObj.stopWindow()
00083
00084     def __dataGetter(self):
00085         """
00086         The __dataGetter function is a private function that gets the data from the CFPB API.
00087         It takes no arguments, but uses self.state_arg and self.year_arg to create a URL for the API call.
00088
00089         Args:
00090             self: Represent the instance of the class
00091
00092         Returns:
00093             A response object
00094
00095         Doc Author:
00096             Willem van der Schans, Trelent AI
00097         """
00098         arg_dict_bu = locals()
00099
00100         link = settings.settingCFBPLink
00101
00102         if self.state_arg is None:
00103             self.state_arg = "UT"
00104         else:
00105             pass
00106
00107         if self.year_arg is None:
00108             self.year_arg = str(datetime.date.today().year - 1)
00109         else:

```

```

00112         pass
00113
00114     passFlag = False
00115
00116     while not passFlag:
00117
00118         self.link = link + f"states={self.state_arg}" + f"&years={self.year_arg}"
00119
00120         response = requests.get(self.link)
00121
00122         if response.status_code == 400:
00123             self.year_arg = int(self.year_arg) - 1
00124
00125         else:
00126             passFlag = True
00127
00128     RESTError(response)
00129     raise SystemExit(0)

```

4.20 ConstructionMonitor/Core.py

```

00001 import copy
00002 import datetime
00003 import json
00004 import os
00005 import threading
00006 import time
00007 from datetime import date, timedelta
00008 from pathlib import Path
00009
00010 import PySimpleGUI as sg
00011 import requests
00012 from cryptography.fernet import Fernet
00013
00014 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00015 from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
00016 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00017 from API_Calls.Functions.DataFunc.Settings import settings
00018 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00019 from API_Calls.Functions.Gui.BatchGui import BatchInputGui
00020 from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
00021 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00022 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00023
00024
00025 class ConstructionMonitorInit:
00026
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         Args:
00034             self: Represent the instance of the class
00035
00036         Returns:
00037             None
00038
00039         Doc Author:
00040             Willem van der Schans, Trelent AI
00041
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
00078     def __ShowGui(self, layout, text):
00079
00080         """
00081         The __ShowGui function is the main function that creates and displays the GUI.
00082         It takes in a layout, which is a list of lists containing all the elements to be displayed on screen.
00083         The text parameter specifies what title should appear at the top of the window.
00084
00085         Args:
00086             self: Refer to the current instance of a class
00087             layout: Determine what the gui will look like
00088             text: Set the title of the window
00089
00090         Returns:
00091             A dictionary of values
00092
00093         Doc Author:
00094             Willem van der Schans, Trelent AI
00095         """
00096         window = sg.Window(text, layout, grab_anywhere=False, return_keyboard_events=True,
00097                             finalize=True,
00098                             icon=ImageLoader("taskbar_icon.ico"))
00099
00100         while True:
00101             event, values = window.read()
00102
00103             if event == "Submit":
00104                 try:
00105                     self.__SetValues(values)
00106                     break
00107                 except Exception as e:
00108                     print(e)
00109                     RESTError(993)
00110                     raise SystemExit(933)
00111             elif event == sg.WIN_CLOSED or event == "Quit":
00112                 break
00113
00114         window.close()
00115
00116     @staticmethod
00117     def __CreateFrame():
00118
00119         """
00120         The __CreateFrame function creates the GUI layout for the application.
00121         The function returns a list of lists that contains all the elements to be displayed in the GUI
00122         window. This is done by creating each line as a list and then appending it to another list which will
00123         contain all lines.
00124
00125         Args:
00126
00127         Returns:
00128             The layout for the gui
00129
00130         Doc Author:
00131             Willem van der Schans, Trelent AI
00132         """
00133         sg.theme('Default1')
00134         line00 = [sg.HSeparator()]

```



```

00135
00136         line0 = [sg.Image(ImageLoader("logo.png")),
00137                 sg.Push(),
00138                 sg.Text("Construction Monitor Utility", font=("Helvetica", 12, "bold"),
justification="center"),
00139                 sg.Push(),
00140                 sg.Push()]
00141
00142         line1 = [sg.HSeparator()]
00143
00144         line3 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00145                 sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"),
key="-Cal-",
00146                         size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-', target="-Cal-")]
00147
00148         line4 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00149                 sg.Input(default_text=date.today().strftime("%Y-%m-%d"), key="-EndCal-",
00150                         size=(20, 1)),
                 sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
00151                                 target="-EndCal-")]
00152
00153         line5 = [sg.HSeparator()]
00154
00155         line6 = [sg.Push(),
00156                 sg.Text("File Settings", font=("Helvetica", 12, "bold"), justification="center"),
00157                 sg.Push()]
00158
00159         line7 = [sg.HSeparator()]
00160
00161         line8 = [sg.Text("Appending File : ", size=(15, None), justification="Right"),
00162                 sg.Input(default_text="", key="-AppendingFile-", disabled=True,
00163                         size=(20, 1)),
                 sg.FileBrowse("Browse File", file_types=[("csv files", "*.csv")], key='-append_file-',
00164                                 target="-AppendingFile-")]
00165
00166         line9 = [sg.HSeparator()]
00167
00168         line10 = [sg.Push(), sg.Submit(focus=True), sg.Quit(), sg.Push()]
00169
00170         layout = [line00, line0, line1, line3, line4, line5, line6, line7, line8, line9, line10]
00171
00172         return layout
00173
00174
00175
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
00183         Args:
00184             self: Represent the instance of the class
00185             values: Pass in the values from the ui
00186
00187         Returns:
00188             A dictionary of values
00189
00190         Doc Author:
00191             Willem van der Schans, Trelent AI
00192         """
00193         self.size = 1000
00194
00195         if values["-Cal-"] != "":
00196             self.dateStart = values["-Cal-"]
00197         else:
00198             self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")
00199
00200         if values["-EndCal-"] != "":
00201             self.dateEnd = values["-EndCal-"]
00202         else:
00203             self.dateEnd = date.today().strftime("%Y-%m-%d")
00204
00205         self.rest_domain = settings.settingCMRestDomain
00206
00207         self.SourceInclude = None
00208
00209         if values["-append_file-"] != "":

```

```

00210         self.append_file = str(values["-append_file-"])
00211     else:
00212         self.append_file = None
00213
00214     self.ui_flag = True
00215
00216
00217 class ConstructionMonitorMain:
00218
00219     def __init__(self, siteClass):
00220
00221         """
00222         The __init__ function is the first function that runs when an object of this class is created.
00223         It sets up all the variables and functions needed for this class to run properly.
00224
00225         Args:
00226             self: Represent the instance of the class
00227             siteClass: Identify the site that is being used
00228
00229         Returns:
00230             Nothing
00231
00232         Doc Author:
00233             Willem van der Schans, Trelent AI
00234
00235         """
00236         self.__siteClass = siteClass
00237         self.__restDomain = None
00238         self.__headerDict = None
00239         self.__columnSelection = None
00240         self.__appendFile = None
00241
00242         self.__parameterDict = {}
00243         self.__search_id = None
00244         self.__record_val = 0
00245         self.__batches = 0
00246
00247         self.__ui_flag = None
00248
00249         self.dataframe = None
00250
00251         try:
00252             self.mainFunc()
00253         except SystemError as e:
00254             if "Status Code = 1000 | Catastrophic Error" in str(getattr(e, 'message', repr(e))):
00255                 print(
00256                     f"ConstructionMonitor/Core.py | Error = {e} | Coerced SystemError in
ConstructionMonitorMain class")
00257                 pass
00258             except AttributeError as e:
00259                 # This allows for user cancellation of the program using the quit button
00260                 if "'NoneType' object has no attribute 'json'" in str(getattr(e, 'message', repr(e))):
00261                     RESTError(1101)
00262                     print(f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Error {e}")
00263                     pass
00264                 elif e is not None:
00265                     print(
00266                         f"ConstructionMonitor/Core.py | Error = {e} | Authentication Error | Please update
keys in AuthUtil")
00267                     RESTError(401)
00268                     print(e)
00269                     pass
00270                 else:
00271                     pass
00272             except Exception as e:
00273                 print(e)
00274                 RESTError(1001)
00275                 raise SystemExit(1001)
00276
00277     def mainFunc(self):
00278         """
00279         The mainFunc function is the main function of this module. It will be called by the GUI or CLI to
execute
00280         the code in this module. The mainFunc function will first create a parameter dictionary using the
__ParameterCreator
00281         method, then it will get a count of all records that match its parameters using the __getCountUI
method, and then
00282         it will calculate how many batches are needed to retrieve all records with those parameters using
BatchCalculator.
00283         After that it asks if you want to continue with retrieving data from Salesforce (if running in GUI
mode). Then it shows

```

```

00284     a progress bar for each
00285
00286     Args:
00287         self: Refer to the current object
00288
00289     Returns:
00290         The dataframe
00291
00292     Doc Author:
00293         Willem van der Schans, Trelent AI
00294     """
00295     self.__ParameterCreator()
00296
00297     print(
00298         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Param Dict =
{self.__parameterDict}")
00299     print(
00300         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Rest Domain =
{self.__restDomain}")
00301
00302     self.__getCountUI()
00303
00304     self.__batches = BatchCalculator(self.__record_val, self.__parameterDict)
00305
00306     print(
00307         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Batches =
{self.__batches} | Rows {self.__record_val}")
00308
00309     if self.__batches != 0:
00310         startTime = datetime.datetime.now().replace(microsecond=0)
00311         eventReturn = BatchInputGui(self.__batches, self.__record_val)
00312         if eventReturn == "Continue":
00313             print(
00314                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Request for
{self.__batches} batches sent to server")
00315             BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00316                                               ParameterDict=self.__parameterDict,
00317                                               HeaderDict=self.__headerDict,
00318                                               ColumnSelection=self.__columnSelection,
00319                                               BatchesNum=self.__batches,
00320                                               Type="construction_monitor")
00321             BatchGuiObject.BatchGuiShow()
00322             self.dataframe = BatchGuiObject.dataframe
00323             print(
00324                 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()))}")
00325             FileSaver("cm", self.dataframe, self.__appendFile)
00326         else:
00327             print(
00328                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Request for
{self.__batches} batches canceled by user")
00329         else:
00330             RESTError(994)
00331             raise SystemExit(994)
00332
00333     def __ParameterCreator(self):
00334         """
00335         The __ParameterCreator function is used to create the parameter dictionary that will be passed into
the
00336         __Request function. The function takes in a siteClass object and extracts all of its attributes,
except for
00337         those that start with '__' or are callable. It then creates a dictionary from these attributes and
stores it as
00338         self.__parameterDict.
00339
00340     Args:
00341         self: Make the function a method of the class
00342
00343     Returns:
00344         A dictionary of parameters and a list of non parameter variables
00345
00346     Doc Author:
00347         Willem van der Schans, Trelent AI
00348     """
00349     __Source_dict = {key: value for key, value in self.__siteClass.__dict__.items() if
00350                     not key.startswith('__') and not callable(key)}
00351
00352     self.__restDomain = __Source_dict["rest_domain"]
00353     __Source_dict.pop("rest_domain")

```

```

00354         self.__headerDict = {"Authorization": __Source_dict["auth_key"]}
00355         __Source_dict.pop("auth_key")
00356         self.__columnSelection = __Source_dict["SourceInclude"]
00357         __Source_dict.pop("SourceInclude")
00358         self.__ui_flag = __Source_dict["ui_flag"]
00359         __Source_dict.pop("ui_flag")
00360         self.__appendFile = __Source_dict["append_file"]
00361         __Source_dict.pop("append_file")
00362
00363         temp_dict = copy.copy(__Source_dict)
00364         for key, value in temp_dict.items():
00365             if value is None:
00366                 __Source_dict.pop(key)
00367             else:
00368                 pass
00369
00370         self.__parameterDict = copy.copy(__Source_dict)
00371
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             __temp_param_dict = copy.copy(self.__parameterDict)
00390             __count_resp = requests.post(url=self.__restDomain,
00391                                         headers=self.__headerDict,
00392                                         json=__temp_param_dict)
00393
00394         except requests.exceptions.Timeout as e:
00395             print(e)
00396             RESTError(790)
00397             raise SystemExit(790)
00398         except requests.exceptions.TooManyRedirects as e:
00399             print(e)
00400             RESTError(791)
00401             raise SystemExit(791)
00402         except requests.exceptions.MissingSchema as e:
00403             print(e)
00404             RESTError(1101)
00405         except requests.exceptions.RequestException as e:
00406             print(e)
00407             RESTError(405)
00408             raise SystemExit(405)
00409
00410         __count_resp = __count_resp.json()
00411
00412         self.__record_val = __count_resp["hits"]["total"]["value"]
00413
00414         del __count_resp, __temp_param_dict
00415
00416     def __getCountUI(self):
00417         """
00418         The __getCountUI function is a wrapper for the __getCount function.
00419         It allows the user to run __getCount in a separate thread, so that they can continue working while it
runs.
00420         The function will display a progress bar and update with text as it progresses through its tasks.
00421
00422         Args:
00423             self: Access the class variables and methods
00424
00425         Returns:
00426             The count of the number of records in the database
00427
00428         Doc Author:
00429             Willem van der Schans, Trelent AI
00430
00431         """
00432

```

```

00433     """
00434     if self.__ui_flag:
00435         uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)
00436
00437         threadGui = threading.Thread(target=self.__getCount,
00438                                     daemon=False)
00439         threadGui.start()
00440
00441         while threadGui.is_alive():
00442             uiObj.textUpdate()
00443             uiObj.windowPush()
00444         else:
00445             uiObj.stopWindow()
00446
00447     else:
00448         self.__getCount()

```

4.21 Realtor/Core.py

```

00001 import datetime
00002 import threading
00003 import time
00004
00005 import pandas as pd
00006 import requests
00007 from bs4 import *
00008
00009 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00010 from API_Calls.Functions.DataFunc.Settings import settings
00011 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00012 from API_Calls.Functions.Gui.BatchGui import confirmDialog
00013 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00014
00015
00016 class realtorCom:
00017
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
00022         anything else in your class.
00023
00024         Args:
00025             self: Represent the instance of the class
00026
00027         Returns:
00028             A new object
00029
00030         Doc Author:
00031             Willem van der Schans, Trelent AI
00032
00033         """
00034         self.__page_html = None
00035         self.__update_date = None
00036         self.__last_date = None
00037         self.__idDict = {"State": "C3", "County": "E3", "Zip": "F3"}
00038         self.__linkDict = {}
00039         self.dfState = None
00040         self.dfCounty = None
00041         self.dfZip = None
00042         self.uiString = "Files Saved to \n"
00043
00044         eventReturn = confirmDialog()
00045         if eventReturn == "Continue":
00046             page_html = requests.get(settings.settingRealtorLink).text
00047             self.__page_html = BeautifulSoup(page_html, "html.parser")
00048             startTime = datetime.datetime.now().replace(microsecond=0)
00049             self.__linkGetter()
00050             print(
00051                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Link Dictionary =
00052                 {self.__idDict}")
00053             self.__showUi()
00054             PopupWrapped(text=self.uiString, windowType="noticeLarge")
00055             print(
00056                 f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Data retrieved with
00057                 in {time.strftime('%H:%M:%S', time.gmtime((datetime.datetime.now().replace(microsecond=0) -
00058                 startTime).total_seconds()))}")
00059         else:

```

```

00055         print(
00056             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | User Canceled
Request")
00057         pass
00058
00059     def __showUi(self):
00060
00061         """
00062         The __showUi function is a helper function that creates and displays the progress window.
00063         It also starts the dataUpdater thread, which will update the progress bar as it runs.
00064
00065
00066         Args:
00067             self: Represent the instance of the class
00068
00069         Returns:
00070             A popupwrapped object
00071
00072         Doc Author:
00073             Willem van der Schans, Trelent AI
00074         """
00075         uiObj = PopupWrapped(text="Request running", windowType="progress", error=None)
00076
00077         threadGui = threading.Thread(target=self.__dataUpdater,
00078                                     daemon=False)
00079         threadGui.start()
00080
00081         while threadGui.is_alive():
00082             uiObj.textUpdate()
00083             uiObj.windowPush()
00084         else:
00085             uiObj.stopWindow()
00086
00087     def __linkGetter(self):
00088
00089         """
00090         The __linkGetter function is a private function that takes the idDict dictionary and adds
00091         a link to each entry in the dictionary. The link is used to access historical data for each
00092         scope symbol.
00093
00094         Args:
00095             self: Refer to the object itself
00096
00097         Returns:
00098             A dictionary of all the links to the history pages
00099
00100         Doc Author:
00101             Willem van der Schans, Trelent AI
00102         """
00103         for key, value in self.__idDict.items():
00104             for row in self.__page_html.find_all("div", {"class": "monthly"}):
00105                 try:
00106                     for nestedRow in row.find_all("a"):
00107                         if "History" in str(nestedRow.get("href")) and key in str(nestedRow.get("href")):
00108                             self.__idDict[key] = {"id": value, "link": nestedRow.get("href")}
00109                 except Exception as e:
00110                     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")
00111                     RESTError(801)
00112                     raise SystemExit(801)
00113
00114     def __dataUpdater(self):
00115
00116         """
00117         The __dataUpdater function is a private function that updates the dataframes for each of the three
00118         types of realtor data. It takes class variables and return the path to the saved file. The
00119         function first creates an empty
00120         dictionary called tempdf, then iterates through each key in self.__idDict (which contains all
00121         three ids).
00122         For each key, it reads in a csv file from the link associated with that id and saves it to tempdf
00123         as a pandas
00124         DataFrame object. Then, depending on which type of realtor data we are dealing with
00125         (State/County/Zip), we save
00126
00127         Args:
00128             self: Access the attributes and methods of the class
00129
00130         Returns:
00131             The path of the saved file

```

```

00130     Doc Author:
00131         Willem van der Schans, Trelent AI
00132     """
00133     for key, value in self.__idDict.items():
00134         tempdf = pd.read_csv(self.__idDict[key]['link'], low_memory=False)
00135
00136         if key == "State":
00137             self.dfState = tempdf
00138         elif key == "County":
00139             self.dfCounty = tempdf
00140         elif key == "Zip":
00141             self.dfZip = tempdf
00142
00143         FileSaveObj = FileSaver(f"realtor_{key}", tempdf)
00144         self.uiString = self.uiString + f"{key} : {FileSaveObj.getPath()} \n"

```

4.22 UtahRealEstate/Core.py

```

00001 import copy
00002 import datetime
00003 import json
00004 import os
00005 import threading
00006 import time
00007 from datetime import date, timedelta
00008 from pathlib import Path
00009
00010 import PySimpleGUI as sg
00011 import requests
00012 from cryptography.fernet import Fernet
00013
00014 from API_Calls.Functions.DataFunc.AuthUtil import AuthUtil
00015 from API_Calls.Functions.DataFunc.BatchProcessing import BatchCalculator
00016 from API_Calls.Functions.DataFunc.FileSaver import FileSaver
00017 from API_Calls.Functions.DataFunc.Settings import settings
00018 from API_Calls.Functions.ErrorFunc.RESTError import RESTError
00019 from API_Calls.Functions.Gui.BatchGui import BatchInputGui
00020 from API_Calls.Functions.Gui.BatchProgressGUI import BatchProgressGUI
00021 from API_Calls.Functions.Gui.ImageLoader import ImageLoader
00022 from API_Calls.Functions.Gui.PopupWrapped import PopupWrapped
00023
00024
00025 class UtahRealEstateInit:
00026
00027     def __init__(self):
00028
00029         """
00030         The __init__ function is called when the class is instantiated.
00031         It sets up the initial state of the object.
00032
00033         Args:
00034             self: Represent the instance of the class
00035
00036         Returns:
00037             The __createframe function
00038
00039         Doc Author:
00040             Willem van der Schans, Trelent AI
00041         """
00042
00043         self.StandardStatus = None
00044         self.ListedOrModified = None
00045         self.dateStart = None
00046         self.dateEnd = None
00047         self.select = None
00048         self.file_name = None
00049         self.append_file = None
00050
00051         self.__ShowGui(self.__CreateFrame(), "Utah Real Estate")
00052
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
00059         __ShowGui

```

```

00059     method then uses these parameters to create an instance of sg.Window with all its attributes set
00060     accordingly.
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
00092     @staticmethod
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"),
00114                    justification="center"),
00115             sg.Push(),
00116             sg.Push()]
00117
00118     line1 = [sg.HSeparator()]
00119
00120     line2 = [sg.Text("MLS Status : ", size=(15, None), justification="Right"),
00121             sg.DropDown(default_value="Active", values=["Active", "Closed"], key="-status-",
00122                    size=(31, 1))]
00123
00124     line3 = [sg.Text("Date Type: ", size=(15, None), justification="Right"),
00125             sg.DropDown(default_value="Listing Date", values=["Listing Date", "Modification Date",
00126                    "Close Date"],
00127                    key="-type-", size=(31, 1))]
00128
00129     line4 = [sg.Text("Start Date : ", size=(15, None), justification="Right"),
00130             sg.Input(default_text=(date.today() - timedelta(days=14)).strftime("%Y-%m-%d"),
00131                    key="-DateStart-",
00132                    disabled=False, size=(20, 1)),
00133             sg.CalendarButton("Select Date", format="%Y-%m-%d", key='-start_date-',
00134                    target="-DateStart-")]
00135
00136     line5 = [sg.Text("End Date : ", size=(15, None), justification="Right"),
00137             sg.Input(default_text=(date.today()).strftime("%Y-%m-%d")), key="-DateEnd-",
00138             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
00159     def __SetValues(self, values):
00160
00161         """
00162         The __SetValues function is used to set the values of the variables that are used in the
00163         __GetData function. The values are passed from a dictionary called 'values' which is created
00164         by parsing through an XML file using ElementTree. This function also sets default values for
00165         some of these variables if they were not specified in the XML file.
00166
00167         Args:
00168             self: Represent the instance of the class
00169             values: Pass the values from the gui to this function
00170
00171         Returns:
00172             A dictionary with the following keys:
00173
00174         Doc Author:
00175             Willem van der Schans, Trelent AI
00176         """
00177         self.StandardStatus = values["-status-"]
00178
00179         self.ListedOrModified = values["-type-"]
00180
00181         if values["-DateStart-"] != "":
00182             self.dateStart = values["-DateStart-"]
00183         else:
00184             self.dateStart = (date.today() - timedelta(days=14)).strftime("%Y-%m-%d")
00185
00186         if values["-DateEnd-"] != "":
00187             self.dateEnd = values["-DateEnd-"]
00188         else:
00189             self.dateEnd = (date.today()).strftime("%Y-%m-%d")
00190
00191         self.select = None
00192
00193         if values["-append_file-"] != "":
00194             self.append_file = str(values["-append_file-"])
00195         else:
00196             self.append_file = None
00197
00198
00199     class UtahRealEstateMain:
00200
00201     def __init__(self, siteClass):
00202
00203         """
00204         The __init__ function is the first function that runs when an object of this class is created.
00205         It sets up all the variables and functions needed for this class to work properly.
00206
00207         Args:
00208             self: Represent the instance of the class
00209             siteClass: Determine which site to pull data from
00210
00211         Returns:
00212             Nothing

```

```

00213
00214 Doc Author:
00215     Willem van der Schans, Trelent AI
00216 """
00217     self.dataframe = None
00218     self.__batches = 0
00219     self.__siteClass = siteClass
00220     self.__headerDict = None
00221     self.__parameterString = ""
00222     self.__appendFile = None
00223     self.__dateStart = None
00224     self.__dateEnd = None
00225     self.__restDomain = settings.settingURERestDomain
00226     self.keyPath = Path(os.path.expandvars(r'%APPDATA%\GardnerUtil\Security')).joinpath(
00227         "3v45wfvw45wvc4f35.av3ra3rvavcr3w")
00228     self.filePath = Path(os.path.expanduser('~\Documents')).joinpath("GardnerUtilData").joinpath(
00229         "Security").joinpath("auth.json")
00230     self.key = None
00231     self.__record_val = None
00232
00233     try:
00234         self.mainFunc()
00235     except KeyError as e:
00236         # This allows for user cancellation of the program using the quit button
00237         if "ListedOrModified" in str(getattr(e, 'message', repr(e))):
00238             RESTError(1101)
00239             print(e)
00240             pass
00241         else:
00242             pass
00243     except Exception as e:
00244         print(e)
00245         RESTError(1001)
00246         raise SystemExit(1001)
00247
00248 def mainFunc(self):
00249     """
00250     The mainFunc function is the main function of this module. It will be called by the GUI when a user
00251     clicks on
00252     the "Run" button in the GUI. The mainFunc function should contain all of your code for
00253     running your program, and it
00254     should return a dataframe that contains all the data you want to display in your final report.
00255
00256     Args:
00257         self: Reference the object itself
00258
00259     Returns:
00260         A dataframe
00261
00262     Doc Author:
00263         Willem van der Schans, Trelent AI
00264     """
00265     passFlag = False
00266     while not passFlag:
00267         if os.path.isfile(self.keyPath) and os.path.isfile(self.filePath):
00268             try:
00269                 f = open(self.keyPath, "rb")
00270                 key = f.readline()
00271                 f.close()
00272                 f = open(self.filePath, "rb")
00273                 authDict = json.load(f)
00274                 fernet = Fernet(key)
00275                 authkey = fernet.decrypt(authDict["ure"]["auth"]).decode()
00276                 self.__headerDict = {authDict["ure"]["parameter"]: authkey}
00277                 passFlag = True
00278             except Exception as e:
00279                 print(
00280                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} |
00281                     UtahRealEstate/Core.py | Error = {e} | Auth.json not found opening AuthUtil")
00282                 AuthUtil()
00283             else:
00284                 AuthUtil()
00285
00286         self.__ParameterCreator()
00287
00288         print(
00289             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')}[:-3]} | Param String =
00290             {self.__parameterString}")
00291         print(

```

```

00290         f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Rest Domain =
(self.__restDomain}")
00291
00292         self.__getCountUI()
00293
00294         if self.__record_val is None:
00295             self.__record_val = 0
00296
00297         self.__batches = BatchCalculator(self.__record_val, None)
00298
00299         print(
00300             f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Batches =
(self.__batches) | Rows {self.__record_val}")
00301
00302         if self.__batches != 0:
00303             startTime = datetime.datetime.now().replace(microsecond=0)
00304             eventReturn = BatchInputGui(self.__batches, self.__record_val)
00305             if eventReturn == "Continue":
00306                 print(
00307                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for
(self.__batches) batches sent to server")
00308                 BatchGuiObject = BatchProgressGUI(RestDomain=self.__restDomain,
00309                                                    ParameterDict=self.__parameterString,
00310                                                    HeaderDict=self.__headerDict,
00311                                                    BatchesNum=self.__batches,
00312                                                    Type="utah_real_estate")
00313                 BatchGuiObject.BatchGuiShow()
00314                 self.dataframe = BatchGuiObject.dataframe
00315                 print(
00316                     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()))}")
00317                 FileSaver("ure", self.dataframe, self.__appendFile)
00318             else:
00319                 print(
00320                     f"{datetime.datetime.today().strftime('%m-%d-%Y %H:%M:%S.%f')[:-3]} | Request for
(self.__batches) batches canceled by user")
00321             else:
00322                 RESTError(994)
00323                 raise SystemExit(994)
00324
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"$filter=ListingContractDate%20gt%20{__Source_dict['dateStart']}%20and%20ListingContractDate%20le%20{__Source_dict['dateEnd']}
00357                 elif __Source_dict["ListedOrModified"] == "Modification Date":
00358                     filter_string =
f"$filter=ModificationTimestamp%20gt%20{__Source_dict['dateStart']}T:00:00:00Z%20and%20ModificationTimestamp%20le%20{__Source_dict['dateEnd']}
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
00366     def __getCount(self):
00367         """
00368         The __getCount function is used to determine the number of records that will be returned by the query.
00369         This function is called when a user calls the count() method on a ReST object. The __getCount function
uses
00370         the $count parameter in OData to return only an integer value representing how many records would be
returned
00371         by the query.
00372
00373         Args:
00374             self: Represent the instance of the class
00375
00376         Returns:
00377             The number of records in the data set
00378
00379         Doc Author:
00380             Willem van der Schans, Trelent AI
00381         """
00382         __count_resp = None
00383
00384         try:
00385             __count_resp = requests.get(f"{self.__restDomain}{self.__parameterString}&$count=true",
00386                                       headers=self.__headerDict)
00387
00388         except requests.exceptions.Timeout as e:
00389             print(e)
00390             RESTError(790)
00391             raise SystemExit(790)
00392         except requests.exceptions.TooManyRedirects as e:
00393             print(e)
00394             RESTError(791)
00395             raise SystemExit(791)
00396         except requests.exceptions.MissingSchema as e:
00397             print(e)
00398             RESTError(1101)
00399         except requests.exceptions.RequestException as e:
00400             print(e)
00401             RESTError(405)
00402             raise SystemExit(405)
00403
00404         self.__record_val = int(__count_resp.json()[["@odata.count"]])
00405
00406     def __getCountUI(self):
00407         """
00408         The __getCountUI function is a wrapper for the __getCount function.
00409         It creates a progress window and updates it while the __getCount function runs.
00410         The purpose of this is to keep the GUI responsive while running long processes.
00411
00412         Args:
00413             self: Represent the instance of the class
00414
00415         Returns:
00416             A popupwrapped object
00417
00418         Doc Author:
00419             Willem van der Schans, Trelent AI
00420         """
00421         uiObj = PopupWrapped(text="Batch request running", windowType="progress", error=None)
00422
00423         threadGui = threading.Thread(target=self.__getCount,
00424                                     daemon=False)
00425         threadGui.start()
00426
00427         while threadGui.is_alive():
00428             uiObj.textUpdate()
00429             uiObj.windowPush()
00430         else:
00431             uiObj.stopWindow()
00432

```