```python
from selenium.webdriver.common.by import By
import logging
import utilities.customlogger as cl
from selenium.webdriver.support.select import Select
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import *
from datetime import date

class SeleniumDriver:
    log = cl.customLogger(logLevel=logging.INFO)

    def __init__(self,driver):
        self.driver = driver

    def getBytype(self,locatortype):
        locator = locatortype.lower()
        if locator == "id":
            return By.ID
        elif locator == "name":
            return By.NAME
        elif locator == "linktext":
            return By.LINK_TEXT
        elif locator == "partiallinktext":
            return By.PARTIAL_LINK_TEXT
        elif locator == "css":
            return By.CSS_SELECTOR
        elif locator == "classname":
            return By.CLASS_NAME
        elif locator == "xpath":
            return By.XPATH
        else:
            self.log.error("Please enter the valid locator :"+locatortype)
            return False

    def getWebElement(self,loctorvalue, locatortype="xpath/id"):
        element = None
        try:
            bytype = self.getBytype(locatortype)
            element = self.driver.find_element(bytype, loctorvalue)
            self.log.info("Identified element with locator type "
                    ""+locatortype+" with locator value "+loctorvalue)
        except Exception as e:
            self.log.error("Element not found "+str(e))
        return element

    def getWebElements(self, loctorvalue, locatortype="xpath/id"):
        listofelements = []
        try:
            bytype = self.getBytype(locatortype)
            listofelements = self.driver.find_elements(bytype, loctorvalue)
            self.log.info("Identified elements with locator type "
                "" + locatortype + " with locator value " + loctorvalue)
        except Exception as e:
            self.log.error("Elements are not found " + str(e))
        return listofelements

    def geturl(self,url):
        self.driver.get(url)
        self.log.info("Entered url "+url)

    def closewindow(self):
        self.log.info("Window closed")
        self.driver.close()

    def quitwindow(self):
```

```
68          self.log.info("All windows closed")
69          self.driver.quit()
70
71      def maximizewindow(self):
72          self.log.info("Window maximized")
73          self.driver.maximize_window()
74
75      def minimizewindow(self):
76          self.log.info("Window minimized")
77          self.driver.minimize_window()
78
79      def setwindowsizeorposition(self,x,y,alter="size/position"):
80          if alter == "size":
81              self.driver.set_window_size(x,y)
82              self.log.info("Setted window size to "+x +"and "+y)
83          elif alter == "position":
84              self.driver.set_window_position(x,y)
85              self.log.info("Setted window position to " + x + "and " + y)
86          else:
87              self.log.error("Please enter the vaid size or position")
88
89      def browsernatives(self,native = "back"):
90          if native == "back":
91              self.driver.back()
92              self.log.info("Clicked on browser native back")
93          elif native == "forward":
94              self.driver.forward()
95              self.log.info("Clicked on browser native forward")
96          elif native == "refresh":
97              self.driver.refresh()
98              self.log.info("Clicked on browser native refresh")
99          else:
100             self.log.error("Please enter the valid native "
101                            "list back/forward/refresh "+native)
102
103     def getsourcecode(self):
104         pagesource = self.driver.page_source
105         self.log.info("Obtained page source is " + pagesource)
106         return pagesource
107
108     def getcurrenturl(self):
109         current_url = self.driver.current_url
110         self.log.info("Obtained current url is " + current_url)
111         return current_url
112
113     def gettitle(self):
114         title = self.driver.title
115         self.log.info("Obtained current title is " + title)
116         return title
117
118     def click(self,locatorvalue,locatortype):
119         try:
120             element = self.getWebElement(locatorvalue,locatortype)
121             element.click()
122             self.log.info("Click on webement with locator type "
123                           +locatortype+ "locator value "+locatorvalue)
124         except Exception as e:
125             self.log.error("Unable to click on webelement with "
126                            "locator type"+locatortype+" locator value "
127                            +locatorvalue+" "+str(e))
128
129     def senddata(self,locatorvalue,locatortype,data):
130         try:
131             element = self.getWebElement(locatorvalue,locatortype)
132             element.send_keys(data)
133             self.log.info("Entered data on webement with locator type "
134                           +locatortype+ "locator value "+locatorvalue+" "+"data = "+data)
```

```
135             except Exception as e:
136                 self.log.error("Unable to enter the data on webelement with "
137                                "locator type"+locatortype+" locator value "
138                                +locatorvalue+" "+str(e))
139 
140         def getText(self,locatorvalue,locatortype):
141             text = None
142             try:
143                 element = self.getWebElement(locatorvalue,locatortype)
144                 text = element.text
145                 self.log.info("Got text from webement with locator type "
146                               +locatortype+ "locator value "+locatorvalue+" "+"text = "+text)
147             except Exception as e:
148                 self.log.error("Unable to get text from webelement with "
149                                "locator type"+locatortype+" locator value "
150                                +locatorvalue+" "+str(e))
151             return text
152 
153         def selectoptionindrpdwn(self,locatorvalue,locatortype,option="India"):
154             try:
155                 element = self.getWebElement(locatorvalue, locatortype)
156                 sel = Select(element)
157                 sel.select_by_visible_text(option)
158                 self.log.info("selcted text from drop down with locator type "
159                               + locatortype + "locator value " + locatorvalue + " " +
                              "option = " + option)
160             except Exception as e:
161                 self.log.error("Unable to select option from drp dwn with "
162                                "locator type" + locatortype + " locator value "
163                                + locatorvalue + " " + str(e))
164 
165         def deselectoptionindrpdwn(self,locatorvalue,locatortype,option="India"):
166             try:
167                 element = self.getWebElement(locatorvalue, locatortype)
168                 sel = Select(element)
169                 sel.deselect_by_visible_text(option)
170                 self.log.info("deselcted text from drop down with locator type "
171                               + locatortype + "locator value " + locatorvalue + " " +
                              "option = " + option)
172             except Exception as e:
173                 self.log.error("Unable to deselect option from drp dwn with "
174                                "locator type" + locatortype + " locator value "
175                                + locatorvalue + " " + str(e))
176 
177         def deselectall(self,locatorvalue,locatortype):
178             try:
179                 element = self.getWebElement(locatorvalue, locatortype)
180                 sel = Select(element)
181                 sel.deselect_all()
182                 self.log.info("deselected all options from drop down with locator type "
183                               + locatortype + "locator value " + locatorvalue + "")
184             except Exception as e:
185                 self.log.error("Unable to deselect all options from drp dwn with "
186                                "locator type" + locatortype + " locator value "
187                                + locatorvalue + " " + str(e))
188 
189         def getfirstselectedoption(self,locatorvalue,locatortype):
190             text = None
191             try:
192                 element = self.getWebElement(locatorvalue, locatortype)
193                 sel = Select(element)
194                 text = sel.first_selected_option.text
195                 self.log.info("Got first selected option from drop down with locator type "
196                               + locatortype + "locator value " + locatorvalue + " " + "text
                              = " + text)
197             except Exception as e:
198                 self.log.error("Unable to obtain first selected option from drp dwn with "
```

```
                            "locator type" + locatortype + " locator value "
                            + locatorvalue + " " + str(e))
        return text

    def mouseover(self,locatorvalue,locatortype):
        try:
            act = ActionChains(self.driver)
            element = self.getWebElement(locatorvalue, locatortype)
            act.move_to_element(element).perform()
            self.log.info("Mouse overed on element with locator type "
                          + locatortype + "locator value " + locatorvalue + "")
        except Exception as e:
            self.log.error("Unable to Mouse over on element with "
                           "locator type" + locatortype + " locator value "
                           + locatorvalue + " " + str(e))

    def leftmouseclick(self,locatorvalue,locatortype):
        try:
            act = ActionChains(self.driver)
            element = self.getWebElement(locatorvalue, locatortype)
            act.move_to_element(element).click().perform()
            self.log.info("Mouse overed on element and clicked with locator type "
                          + locatortype + "locator value " + locatorvalue + "")
        except Exception as e:
            self.log.error("Unable to Mouse over on element and click with "
                           "locator type" + locatortype + " locator value "
                           + locatorvalue + " " + str(e))

    def scrolltoptobottom(self,x=0,y=1000):
        self.driver.execute_script("window.scrollBy("+str(x)+","+str(y)+");")
        self.log.info("Scrolled the page from top to bottom "+str(x)+ " "+str(y))

    def scrollbottomtotop(self,x=0,y=-500):
        self.driver.execute_script("window.scrollBy("+str(x)+","+str(y)+");")
        self.log.info("Scrolled the page from bottom to top "+str(x)+ " "+str(y))

    def switchtoframe(self,id=None,index=0):
        try:
            if id is not None:
                self.driver.switch_to.frame(id)
                self.log.info("Switched into the frame with id "+id)
            else:
                self.driver.switch_to.frame(index)
                self.log.info("Swicted into the frame with index "+index)
        except Exception as e:
            self.log.error("Unable to switch to frame "+str(e))

    def switchtoparentframe(self):
        self.driver.switch_to.parent_frame()


    def waitforelementclickable(self,locatorvalue,locatortype,time=60,poll=10):
        try:
            bytype = self.getBytype(locatortype)
            wait = WebDriverWait(self.driver, time, poll_frequency=poll,
                                 ignored_exceptions=[NoSuchElementException,
                                                     ElementNotInteractableException])
            wait.until(EC.element_to_be_clickable((
                bytype, locatorvalue)))
            self.log.info("Waited for element to be clicked")
        except Exception as e:
            self.log.error("Waited for element to be clicked time = "+str(time)+" But
            unsccuessfull")

    def waitforvlisbleofelement(self,locatorvalue,locatortype,time=60,poll=10):
        try:
            bytype = self.getBytype(locatortype)
```

```
            wait = WebDriverWait(self.driver, time, poll_frequency=poll,
                                 ignored_exceptions=[NoSuchElementException,
                                                     ElementNotInteractableException])
            wait.until(EC.visibility_of((
                bytype, locatorvalue)))
            self.log.info("Waited for visble of element")
        except Exception as e:
            self.log.error("Waited for element to be visible time = "+str(time)+" But
            unsccuessfull")


    def elementisselcted(self,locatorvalue,locatortype):
        element = self.getWebElement(locatorvalue, locatortype)
        return element.is_selected()

    def elementisenabled(self, locatorvalue, locatortype):
        element = self.getWebElement(locatorvalue, locatortype)
        return element.is_enabled()

    def elementisdisplayed(self, locatorvalue, locatortype):
        element = self.getWebElement(locatorvalue, locatortype)
        return element.is_displayed()

    def getcurrentwindowid(self):
        currentwindowid = self.driver.current_window_handle
        self.log.info("Obtained window id = "+currentwindowid)
        return currentwindowid

    def getallwindowids(self):
        allwindowids = self.driver.window_handles
        self.log.info("Obtained all window ids = " + allwindowids)
        return allwindowids

    def switchtowindow(self,windowid):
        self.driver.switch_to.window(windowid)
        self.log.info("Switched into window with window id = "+windowid)


    def switchtoJSpopup(self,action="accept/dismiss"):
        if action == "accept":
            self.driver.switch_to.alert.accept()
            self.log.info("Clicked on OK button")
        elif action == "dismiss":
            self.driver.switch_to.alert.dismiss()
            self.log.info("Clicked on CANCEL button")
        else:
            self.log.error("Please provide the valid action")

    def gettextfromJSpopup(self):
        return self.driver.switch_to.alert.text

    def getcurrentdate(self):
        today = date.today()
        currentdate = today.strftime("%d")
        self.log.info("Current date is "+currentdate)
        return currentdate
```