```python
import time

from urllib import parse
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager

from Logger import Logging

# For selenium driver implementation on heroku
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--disable-gpu')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument("disable-dev-shm-usage")

logger_obj = Logging('Advance Image Downloader')  # Creating a custom based logger
logger_obj.initialize_logger()  # Instantiating the logger object


class ImageScrapperClass:

    def __init__(self, no_images):
        """
        This function will instantiate the query email and no_images parameter
        :param no_images: Number of images want
        """
        try:
            self.no_images = no_images
            self.browser = webdriver.Chrome(executable_path=ChromeDriverManager().install(),
                                            chrome_options=chrome_options)
        except Exception as e:
            logger_obj.print_log('(Scrapper.py(__init__) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def get_request(self, search_query):
        """
        This function will open the chrome browser for fetching operations
        """
        try:
            # Parsing the search query for searching over google
            search_query = parse.quote_plus(search_query)

            url = 'https://www.google.com/search?hl=jp&q=' + search_query + '&btnG=Google+Search&tbs=0&safe=off&tbm' \
                                                          '=isch '
            self.browser.get(url)
        except Exception as e:
            logger_obj.print_log('(Scrapper.py(get_request) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def store_url(self, thumbnail_images, current_count, no_thumbnail_images, final_images,
req_id, email, cassandra):
        """This function inserts the images URL inside the database and then returns the this URL in the set format
        :param final_images: Images URL found
        :param thumbnail_images: Thumbnails selected
        :param current_count: Current count of urls found
        :param no_thumbnail_images: Number of images selected
        :param cassandra: Cassandra object
        :param email: Email ID of user
        :param req_id: Unique Request ID
        :return: set of url found
        """
        try:
            task_finished = False
            for img in thumbnail_images[current_count: no_thumbnail_images]:
                # Try to click on every thumbnail so that it can open at side
                try:
                    img.click()
                    time.sleep(0.5)
                except Exception as e:
                    logger_obj.print_log('(Scrapper.py(store_url)) - Something went wrong ' + str(e), 'info')
                    continue

                # Extract the image urls from the thumbnail which is opened
                opened_images = self.browser.find_elements_by_css_selector('img.n3VNCb')
                for actual_image in opened_images:
                    if actual_image.get_attribute('src') and 'http' in actual_image.get_attribute('src'):
                        if not actual_image.get_attribute('src') in final_images:
                            final_images.add(actual_image.get_attribute('src'))
                            cassandra.insert_url(req_id, email, actual_image.get_attribute('src'))
                            current_count += 1

                # Condition to not return the more images then what is expected from the user
                if current_count >= self.no_images:
                    task_finished = True
                    break

            return final_images, current_count, task_finished
        except Exception as e:
            logger_obj.print_log('(Scrapper.py(store_url) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def fetch_thumbnails(self, req_id, email, cassandra):
        """
        This function will fetch the thumbnails of images from the Chrome
        :param email: Email ID of the user
        :param cassandra: Cassandra object
        :param req_id : Unique request id of the request
        """
        try:
            final_images = set()
            current_count = 0
            task_finished = False

            while current_count < self.no_images and not task_finished:
                # Scroll to the end of the result section
                self.scroll_to_end()

                # If show more button exists then click
                if self.browser.find_element_by_css_selector('.mye4qd'):
                    self.browser.execute_script('document.querySelector(".mye4qd").click()')

                # Get all the thumbnail result
                thumbnail_images = self.browser.find_elements_by_css_selector('img.Q4LuWd')
                no_thumbnail_images = len(thumbnail_images)

                final_images, current_count, task_finished = self.store_url(thumbnail_images, current_count,
                                                                             no_thumbnail_images, final_images, req_id,
                                                                             email, cassandra)

            logger_obj.print_log('Images URL has been downloaded', 'info')

            # Writing all the links in to the files
            print('The number of images are {}'.format(len(final_images)))

            # Closing the browser connection
            self.close_browser()

        except Exception as e:
            logger_obj.print_log('(Scrapper.py(fetch_thumbnails) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def scroll_to_end(self):
        """
        This function scrolls the browser window to the end of the document body
        """
        try:
            self.browser.execute_script('window.scrollTo(0, document.body.scrollHeight);')
            time.sleep(1.5)
        except Exception as e:
            logger_obj.print_log('(Scrapper.py(scroll_to_end) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def close_browser(self):
        """
        This function closes the browser object
        """
        try:
            self.browser.close()
        except Exception as e:
            logger_obj.print_log('(Scrapper.py(close_browser) - Something went wrong ' + str(e
), 'exception')
            raise Exception(e)

```