```python
from config import email_config
from Cassandra import Cassandra
from Download import Download
from Scrapper import ImageScrapperClass
from Logger import Logging
from Email import SendEmail

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


class HelperClass:

    def helper_image(self, search_query, no_images, email, req_id, schedule_job):
        """
        this helper method responsible for calling the respective methods for inserting the images into the database
        :param search_query: Search query given by user
        :param no_images: Number of images to download
        :param email: Email of the user
        :param req_id: Unique request ID of the user
        :param schedule_job: scheduler object
        """
        try:
            # Send email about job starts execution
            message = 'Subject: {}\n\n{}'.format('Job scheduling started',
                                                 'Your job has been started and it might take few minutes to complete.'
                                                 ' You will get confirmation and the download link in the mail once the'
                                                 ' process is completed.')

            self.helper_email(email, str(req_id), search_query, message)

            # Initializing the cassandra database object
            cassandra = Cassandra()

            # Connecting to the default keyspace
            cassandra.connect_keyspace()

            # Creating the table if not exists
            cassandra.create_table()

            # Initializing the image_scrapper for web scrapping
            image_scrapper = ImageScrapperClass(no_images)

            # Opening the URL provided in the Chrome tab
            image_scrapper.get_request(search_query)

            # Storing the URL
            image_scrapper.fetch_thumbnails(req_id, email, cassandra)

```

```
50          self.helper_download(email, search_query, req_id, schedule_job, cassandra)
51
52      except Exception as e:
53          logger_obj.print_log('(HelperClass.py(helper_image) - Something went wrong ' + str
    (e), 'exception')
54
55          # Sending the error message
56          error_message = 'Subject: Error in job\n\nThere is some error occurred while
    performing your job ' \
57                          'activities. Would you like to retry again? \n' + email_config.host_name
58          self.helper_email(email, message=error_message)
59
60          # Deleting the URL and files which are inserted/created
61          self.helper_delete(str(req_id))
62          raise Exception(e)
63
64    @staticmethod
65    def helper_email(email, req_id=None, search_query=None, message=None):
66        """
67        This helper method is responsible for calling methods for sending an email
68        :param email: Email address of  the receiver
69        :param reqs_id: Unique request ID
70        :param search_query: Search query of the user
71        :param message: Message to be sent
72        """
73        try:
74            if not message:
75                message = 'Subject: Your Images are Downloaded\n\nKindly download your
    images through the following ' \
76                          'link. Link is valid for {} minutes\n {}'.format(
77                    email_config.time_to_delete_min,
78                    email_config.host_name + 'download/' + search_query.replace(' ', '') + '/' + str(
    req_id))
79
80          # Initializing the email object
81          email_obj = SendEmail()
82          # Sending the notification
83          email_obj.send_notification(email, message)
84
85      except Exception as e:
86          logger_obj.print_log('(HelperClass.py(helper_email) - Something went wrong ' + str(
    e), 'exception')
87          raise Exception(e)
88
89    @staticmethod
90    def helper_delete(req_id):
91        """
92        This helper method is responsible for calling the methods to delete the files after some
    amount of time
93        :param req_id: Unique request ID
94        """
95        try:
```

```
 96            print("Delete operation started")
 97            logger_obj.print_log('Deleting operation started', 'info')
 98
 99            # Deleting the files from the system
100            Download.delete_file(req_id)
101
102            logger_obj.print_log('All the files are deleted', 'info')
103            print("All the files are deleted now")
104
105            # Initializing the cassandra database object
106            cassandra = Cassandra()
107            logger_obj.print_log('Connected to the cassandra', 'info')
108            print('Connected to the cassandra')
109            # Connecting to the default keyspace
110            cassandra.connect_keyspace()
111            # Deleting the database records
112            cassandra.delete_url(req_id)
113            logger_obj.print_log('Delete operation from cassandra is done', 'info')
114            print('Delete operation from the cassandra is done')
115
116        except Exception as e:
117            logger_obj.print_log('(HelperClass.py(helper_delete) - Something went wrong ' + str
    (e), 'exception')
118            raise Exception(e)
119
120    def helper_download(self, email, search_query, req_id, schedule_job, cassandra=None):
121        """
122        This helper method is responsible for calling the methods to download the images over
    an internet
123        :param email: Email address of  the receiver
124        :param search_query: Search query of the user
125        :param req_id: Unique request ID
126        :param schedule_job: SchedulerClass object
127        :param cassandra: Cassandra object
128        """
129        try:
130            logger_obj.print_log('Inside the download function', 'info')
131
132            # Initializing the cassandra database object
133            if not cassandra:
134                cassandra = Cassandra()
135                # Connecting to the default keyspace
136                cassandra.connect_keyspace()
137                logger_obj.print_log('Connection to the database established', 'info')
138
139            # First check whether the req_id exists in the database
140            result = cassandra.select_query(req_id)
141
142            if result:
143                logger_obj.print_log('Result is found', 'info')
144
145                # Creating the download class object for performing download operations
```

```
146             download_obj = Download(result)
147
148             str_req_id = str(req_id)
149
150             # Creating directory for saving images
151             Download.create_dir(str_req_id)
152             logger_obj.print_log('Directory is created', 'info')
153
154             # Downloading Images
155             download_obj.download_images(search_query, str_req_id)
156             logger_obj.print_log('Images are downloaded over the internet', 'info')
157
158             # Creating a zip file
159             Download.create_zip(str_req_id)
160             logger_obj.print_log('Zip file is created', 'info')
161
162             # Calling the delete_files function
163             schedule_job.delete_files_job_queue(str_req_id, email_config.
    time_to_delete_min)
164
165             # Sending the mail
166             self.helper_email(email, str_req_id, search_query)
167         else:
168             logger_obj.print_log('(app.py) - Something went wrong You are not allowed to
    access', 'exception')
169             raise Exception('You are not allowed to access or the Link has expired')
170
171     except Exception as e:
172         logger_obj.print_log('(app.py) - Something went wrong You are not allowed to
    access', 'exception')
173         raise Exception(e)
174
```