```python
# Importing the necessary imports
import re
import uuid
from flask import Flask, render_template, request, send_file
from flask_cors import cross_origin

from Scheduler import ScheduleJob
from Logger import Logging

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

app = Flask(__name__)  # Initializing the Flask App with the name 'app'


# Home Page Route
@app.route('/', methods=['GET'])
@cross_origin()
def index():
    """
    Function is responsible for showing the index page
    """
    try:
        if request.method == 'GET':
            logger_obj.print_log('Inside the index function', 'info')
            logger_obj.print_log('Rendering the index.html template', 'info')
            return render_template('index.html')
        else:
            logger_obj.print_log('(app.py) - Something went wrong Method not allowed', '
exception')
            return render_template('error.html', msg='Method not allowed')
    except Exception as e:
        logger_obj.print_log('(app.py) - Something went wrong ' + str(e), 'exception')
        return render_template('error.html', msg=str(e))


#  Submitted Page Route
@app.route('/job_submitted', methods=['POST'])
@cross_origin()
def job_submitted():
    """
    The function is responsible for performing the various actions after the job is submitted by
    the user
    """
    try:
        if request.method == 'POST':
            logger_obj.print_log('Inside the job_submitted function', 'info')

            # Handling the user input
            search_query = request.form['search-query'].lower()
            date = request.form['date']
```

```
51          time = request.form['time']
52          email = request.form['email'].lower()
53          no_images = request.form['images']
54
55          is_valid, error = validate_inputs(search_query, date, time, email, no_images)
56
57          if is_valid:
58              # Creating the unique ID for the request generated
59              req_id = uuid.uuid4()
60
61              # Creating a object for th e scheduler
62              schedule_job = ScheduleJob()
63
64              # Adding the job in the scheduler
65              schedule_job.insert_request(search_query, date, time, int(no_images), email,
    req_id)
66
67              logger_obj.print_log('Schedule is added for adding the job in queue', 'info')
68
69              # Rendering the Job Submitted template
70              logger_obj.print_log('Rendering the job_submitted.html template', 'info')
71              return render_template('job_submitted.html')
72          else:
73              logger_obj.print_log('(app.py) - Something went wrong ' + error, 'exception')
74              return render_template('error.html', msg=error)
75
76      else:
77          logger_obj.print_log('(app.py) - Something went wrong Method is not allowed', '
    exception')
78          return render_template('error.html', msg='Method not allowed')
79
80  except ValueError:
81      logger_obj.print_log('(app.py) - Something went wrong. No of images must be a
    number', 'exception')
82      return render_template('error.html', msg='No of images must be a number')
83
84  except Exception as e:
85      logger_obj.print_log('(app.py) - Something went wrong ' + str(e), 'exception')
86      return render_template('error.html', msg=str(e))
87
88
89  # Downloading the images route
90  @app.route('/download/<search_term>/<uuid:req_id>', methods=['GET'])
91  @cross_origin()
92  def download(search_term, req_id):
93      """
94      Function is responsible for sending the zip file to the user
95      :param search_term: Search query of the user
96      :param req_id: Unique request ID of the user
97      :return: Zip file created
98      """
99      try:
```

```
100         logger_obj.print_log('Inside the download route', 'info')
101         str_req_id = str(req_id)
102 
103         # Sending the downloadable file to the user
104         return send_file(str_req_id + '_zipfile.zip', as_attachment=True, attachment_filename=
    search_term + '.zip')
105 
106     except Exception as e:
107         logger_obj.print_log('(app.py) - Something went wrong ' + str(e), 'exception')
108         return render_template('error.html', msg='This link has expired')
109 
110 
111 def validate_inputs(search_query, date, time, email, no_images):
112     """
113     Function is responsible for validating the inputs given by the user
114     :param search_query: search term by the user
115     :param date: Date for scheduling the job
116     :param time: Time for scheduling the job
117     :param email: Email address of the user
118     :param no_images: No of images given by the user
119     :return: Boolean if the input's are valid
120     """
121     try:
122         # Checking if the queries passed are empty
123         if search_query != '' and date != '' and time != '' and email != '' and no_images != '':
124 
125             no_images = int(no_images)  # Converting into integer for further processing
126             # Number of images should be in between 1 and 500
127             if 1 <= no_images <= 500:
128                 # Validating the email address
129                 if re.search('^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$', email):
130                     return True, None
131                 else:
132                     logger_obj.print_log('(app.py (validate_inputs)) - Something went wrong. Email
     address is invalid',
133                                          'exception')
134                     return False, 'Invalid email address'
135             else:
136                 logger_obj.print_log('(app.py (validate_inputs)) - Something went wrong. No of
    images must be in '
137                                      'between 1 and 500',
138                                      'exception')
139                 return False, 'No of images must be in between 1 and 500'
140         else:
141             logger_obj.print_log('(app.py (validate_inputs)) - Something went wrong. One of
    the inputs is empty',
142                                  'exception')
143             return False, 'One of inputs is empty'
144 
145     except Exception as e:
146         raise Exception(e)
147 
```

```
148
149 if __name__ == '__main__':
150     app.debug = True
151     app.run()
152
```