Software Engineering
Group submission
Lab 8 Group - 3
Done by Kandarp Devmurari(202001052) only

Testcases for python code:

Testcase - 1:

Sending an empty string for upload folder.

```
app.secret_key = "secret key"
# UPLOAD_FOLDER = r"static\upload"
UPLOAD_FOLDER = r""
```

Error:

TypeError

TypeError: The view function for 'upload_file' did not return a valid response. The function either returned None or ended without a return statement.

Traceback (most recent call last)

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *2088*, in __call__

```
return self.wsgi_app(environ, start_response)
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *2073*, in wsgi_app

```
response = self.handle_exception(e)
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *2070*, in wsgi_app

```
response = self.full_dispatch_request()
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *1516*, in full_dispatch_request

```
return self.finalize_request(rv)
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *1535*, in finalize_request

```
response = self.make_response(rv)
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *1698*, in make_response

```
raise TypeError(
```

TypeError: The view function for 'upload_file' did not return a valid response. The function either returned None or ended without a return statement.

Testcase -2

Sending an invalid string for upload folder.

```
UPLOAD_FOLDER = r"s\upload"
```

Error:

FileNotFoundError

FileNotFoundError: [Errno 2] No such file or directory: 's\\upload\\cars.mp4'

Traceback (most recent call last)

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *2088*, in __call__

```
return self.wsgi_app(environ, start_response)
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *2073*, in wsgi_app

```
response = self.handle_exception(e)
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *2070*, in wsgi_app

```
response = self.full_dispatch_request()
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *1515*, in full_dispatch_request

```
rv = self.handle_user_exception(e)
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *1513*, in full_dispatch_request

```
rv = self.dispatch_request()
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\flask\app.py", line *1499*, in dispatch_request

```
return self.ensure_sync(self.view_functions[rule.endpoint])(**req.view_args)
```

File "C:\Users\kanda\Desktop\github_speed_algo_2\V-core\app.py", line *55*, in upload_file

```
file.save(os.path.join(app.config["UPLOAD_FOLDER"], filename))
```

File "C:\Users\kanda\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\werkzeug\datastructures.py", line *3013*, in save

```
dst = open(dst, "wb")
```

FileNotFoundError: [Errno 2] No such file or directory: 's\\upload\\cars.mp4'

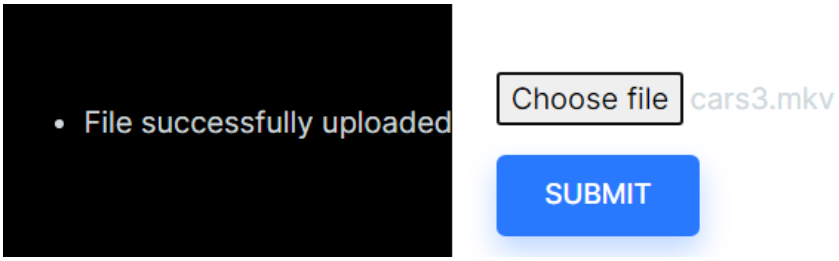
Testcase - 3

Sending valid input folder string for upload folder.

```
UPLOAD_FOLDER = r"static\upload"
```

Error:

No error file get uploaded successfully



Therefore, the testcases 1,2 can be fixed by entering the valid folder to contain the uploaded video.

Testcase - 4:

Sending an empty xml file string for dataset1

```
dataset_1 = cv2.CascadeClassifier(r"")
dataset_2 = cv2.CascadeClassifier(r"dataset\myhaar.xml")
```

Error:

back

Video is not displayed, and hence no object detection takes place.

Testcase - 5

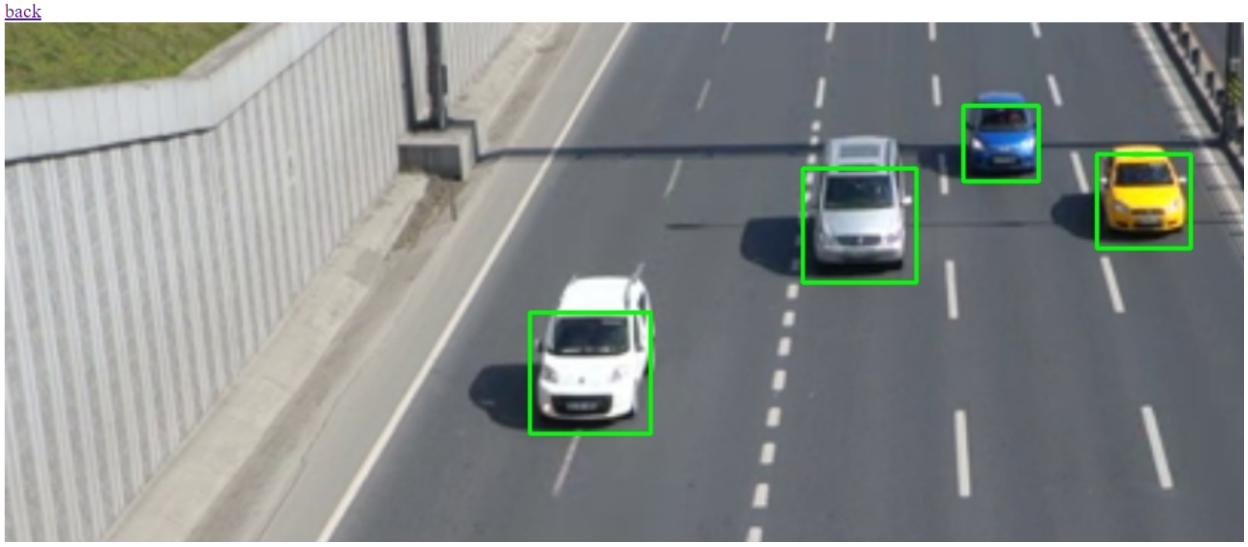
Sending valid input xml file string for dataset1.

```
dataset_1 = cv2.CascadeClassifier(r"dataset\cars.xml")
dataset_2 = cv2.CascadeClassifier(r"dataset\myhaar.xml")
```

Error:

No error, video displayed successfully and object detection/tracking takes place

back



Therefore, the testcase 4 can be fixed by entering the valid xml file location for the dataset1.

Testcase - 6, 7: can be made similarly as testcases - 4,5 for empty and valid xml file strings for dataset2(which is not actually used for tracking, but only used for quality checking for detection and tracking). These also give the same error/success output as in testcases - 4,5

—----------X—----------X—----------

Testcases for the javascript code:

1. Input:


```
{
  "email": "test@example.com",
  "password": "test123"
}
```


Expected Output:


```
{
  "message": "User created successfully",
  "user": {
    "email": "test@example.com",
    "password": "test123",
    "_id": "60f57d6f1db7c12a9c30e6e1",
    "createdAt": "2021-07-19T06:55:27.824Z",
    "__v": 0
  }
}
```


2. Input:


```
{
  "email": "test@example.com",
  "password": "test123"
}
```

Expected Output:

```
{
  "message": "User already exist"
}
```

3. Input:

```
{
  "email": "testexample.com",
  "password": "test123"
}
```

Expected Output:

```
{
  "message": "Please enter a valid email"
}
```

4. Input:

```
{
  "email": "",
  "password": ""
}
```

Expected Output:

```
{
  "message": "Please enter email and password"
}
```

Executing the test cases

The JUnit testing framework is typically used for Java code, and may not be suitable for testing Node.js applications written in JavaScript. Instead, we can

use a Node.js testing framework like Mocha, along with a testing library like Chai or Jest.

We can create a test file, let's call it app.test.js. This file should be placed in the same directory as the app.js file.

```
const chai = require('chai');
const chaiHttp = require('chai-http');
const expect = chai.expect;
const app = require('./app');

chai.use(chaiHttp);

describe('API Tests', () => {
 // Test for root route
 describe('GET /', () => {
  it('should return "Hello World!"', (done) => {
   chai.request(app)
    .get('/')
    .end((err, res) => {
     expect(res).to.have.status(200);
     expect(res.text).to.equal('Hello World!');
     done();
    });
  });
 });

 // Test for register route
 describe('POST /register', () => {
  it('should create a new user and return a success message', (done) => {
   chai.request(app)
    .post('/register')
    .send({
     email: 'test@example.com',
```

```
        password: 'password123'
      })
      .end((err, res) => {
        expect(res).to.have.status(200);
        expect(res.body.message).to.equal('User created successfully');
        expect(res.body.user.email).to.equal('test@example.com');
        done();
      });
  });

  it('should return an error if email or password is missing', (done) => {
    chai.request(app)
      .post('/register')
      .send({})
      .end((err, res) => {
        expect(res).to.have.status(400);
        expect(res.text).to.equal('Please enter email and password');
        done();
      });
  });

  it('should return an error if email is invalid', (done) => {
    chai.request(app)
      .post('/register')
      .send({
        email: 'invalid-email',
        password: 'password123'
      })
      .end((err, res) => {
        expect(res).to.have.status(400);
        expect(res.text).to.equal('Please enter a valid email');
        done();
      });
  });
```

```
  it('should return an error if user already exists', (done) => {
    chai.request(app)
      .post('/register')
      .send({
        email: 'test@example.com',
        password: 'password123'
      })
      .end((err, res) => {
        expect(res).to.have.status(400);
        expect(res.text).to.equal('User already exist');
        done();
      });
  });
});

// Test for login route
describe('POST /login', () => {
  it('should log in the user and return a success message', (done) => {
    chai.request(app)
      .post('/login')
      .send({
        email: 'test@example.com',
        password: 'password123'
      })
      .end((err, res) => {
        expect(res).to.have.status(200);
        expect(res.body.message).to.equal('Login successful');
        expect(res.body.user.email).to.equal('test@example.com');
```