API Documentation

Contents

auth.js

Contains register and login routes, Located at **routes/auth.js**

This file contains the routes for the authentication (registration, login)

Sample Call in React

```
    axios.post('/auth/register', { name, email, password, role })
      .then((result) => {
        this.props.history.push("/login")
      });
  }
```

Register New User Route

/auth/register

Description:

The User creates a new account by entering the required fields (name, email, password, role). If the User is successful, a new id is created in the **user model.** The "role" is an enumerator value which is important for determining the role-based authorization throughout the application. The User has to enter a specified role ("employee", "employer", "admin", "institute"). These roles should be included in a drop-down or radial form selection.

The user should then be redirected to the dashboard, homepage or to one of the following routes where the user can add personal details corresponding to their chosen role:

- **/employees/add/:id**
- **/employers/add/:id**
- **/admin/add/:id**
- **/institution/add/:id**

The id created from the successful registration will be included as a parameter.

URL Params

Required:

```
{
      "name": "usersname",
      "email": "user@email",
      "password": "password",
      "role": "employee"
  }
```

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "success": true,
  "msg": "New User Created Successfully."
}
```

- **Code:** 200

  Content:

```
{
  "success": false,
  "msg": "Email already exists."
}
```

- **Code:** 200

  Content:

```
{
  "success": false,
  "msg": "Username already exists."
}
```

- **Code:** 200

  Content:

```
{
  "success": false,
  "msg": "Email already exists."
}
```

Error Response:

- **Code:** 400 Bad Request

  Content:

```
{
  "name": “Name field is required”
}
```

- **Code:** 400 Bad Request
  Content:

```
{
  "Email": “Email field is required”
}
```

- **Code:** 400 Bad Request
  Content:

```
{
  "password": “Password must be at least 6 characters”
}
```

User Login Route

/auth/login

Method: Post

URL Params:

Required

```
{
      "name": "usersname",
      "password": "password"
  }
```

Sample Call:

```
axios.post('/auth/login', { name, password })
            .then((result) => {
                localStorage.setItem('jwtToken', result.data.token);
                this.setState({ message: '' });
                this.props.history.push('/')
            })
      .catch((error) => {
        if(error.response.status === 401) {
   this.setState({ message: 'Login failed. Username or password not match' });
        }
      });
```

```
    }
```

Data Params:

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "success": true,
  "token": "JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9”
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "success": false,
  "msg": "Authentication failed. User not found."
}
```

- **Code:** 401 Unauthorized

  Content:

```
{
  "success": false,
  "msg": "Authentication failed. Wrong password. "
}
```

models\user.js

```
var UserSchema = new Schema({
    name: {
        type: String,
        unique: true,
        required: true
    },
    email: {
        type: String,
        unique: true,
        required: true
    },
    password: {
        type: String,
        required: true
    },
```

```
    role: {
        type: String,
        enum: ['admin', 'employer', 'employee', 'institute'],
        required: true
    },
    date: {
        type: Date,
        default: Date.now
    },
      followed: [followSchema],
    employment:[workersSchema],


    employeeDetails:[employeeSchema],
    employeeCerts:[empCertsSchema],

    employerDetails:[employerSchema],
    insurance:[insurSchema],

    assets:[assetSchema]
    ,
    assetCerts:[certSchema]
    ,
    safety_statements:[safeSchema],

    adminDetails:[adminSchema],

    instituteDetails:[instituteSchema],
    e_learning:[e_LearnSchema]


});
```

users.js

Followed routes

Make a request to follow a user:

Description:

A logged in User can make a request to follow another user for the purpose of gaining read/ write access to their content.

- User must be logged in

UserA id is pushed to UserB as a request pending further action

```
// User B (recipient) result
```

```
"_id": "5e9efe47263eb40fd0125b3b ",

"followed": [
    {
      "_id": "5e85bd74f9fd5c113013fb96",
      "requester": "5e7f8e4202a95532bce85947",
      "status": "pending",
      "updatedAt": "2020-04-02T10:24:52.289Z",
      "createdAt": "2020-04-02T10:24:52.289Z"
    }
```

UserA status shows that a request has been made to the recipient

```
// User A (requester) result

"_id": "5e7f8e4202a95532bce85947",

"followed": [
    {
      "_id": "5e85bd74f9fd5c113013fb96",
      "recipient": "5e9efe47263eb40fd0125b3b",
      "status": "requested",
      "updatedAt": "2020-04-02T10:24:52.289Z",
      "createdAt": "2020-04-02T10:24:52.289Z"
    }
```

URL

/users/requestMake/:UserA/:UserB

Method

PUT

URL Params

:UserA = [string], :UserB = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
  {
    {
  "_id": "5e9e1d6be2ecca2b58ea91a1",
  "name": "austin",
  "email": "austin@email.com",
```

```
  "password": "$2a$10$IkAGHojyrTwbvCCgNoy0E.3b1AnVnFQTD5Ozppay41M.EEhEoZM5q",
  "role": "employer",
  "date": "2020-04-20T22:08:43.273Z",
  "followed": [],
  "employment": [],
  "employeeDetails": [],
  "employeeCerts": [],
  "employerDetails": [],
  "insurance": [],
  "assets": [],
  "assetCerts": [],
  "safety_statements": [],
  "adminDetails": [],
  "intsituteDetails": [],
  "e_learning": [],
  "__v": 0
} [
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
  "stringValue": "\"{ recipient: \\'5e9f71816a75f728b081\\',\\n  status:
\\'requested\\',\\n  updatedAt: 2020-04-27T13:10:44.996Z,\\n  createdAt: 2020-
04-27T13:10:44.996Z }\"",
  "kind": "embedded",
  "value": "{ recipient: '5e9f71816a75f728b081',\n  status: 'requested',\n
updatedAt: 2020-04-27T13:10:44.996Z,\n  createdAt: 2020-04-27T13:10:44.996Z
}",
  "path": "followed",
  "reason": {
    "stringValue": "\"5e9f71816a75f728b081\"",
    "kind": "ObjectID",
    "value": "5e9f71816a75f728b081",
    "path": "recipient",
    "reason": {
      "stringValue": "\"5e9f71816a75f728b081\"",
     "kind": "ObjectId",
      "value": "5e9f71816a75f728b081",
      "path": "recipient",
      "reason": {},
      "message": "Cast to ObjectId failed for value \"5e9f71816a75f728b081\"
at path \"recipient\"",
      "name": "CastError"
    },
    "message": "Cast to ObjectID failed for value \"5e9f71816a75f728b081\" at
path \"recipient\"",
```

```
    "name": "CastError"
  },
  "message": "Cast to embedded failed for value \"{ recipient:
\\'5e9f71816a75f728b081\\',\\n  status: \\'requested\\',\\n  updatedAt: 2020-
04-27T13:10:44.996Z,\\n  createdAt: 2020-04-27T13:10:44.996Z }\"at path \"fol-
lowed\"",
  "name": "CastError"}
```

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

Reject a request from a user:

Description:

A logged in User can reject a request from another user, This action will pull the Followed data created during the /users/requestMake/:UserA/:UserB call from the database

- User must be logged in

URL

/users/reject/:UserA/:UserB

Method

PUT

URL Params

:UserA = [string], :UserB = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
```

```
  {
  "_id": "5e7f8e5202a95532bce85948",
  "name": "shane cunningham",
  "email": "shane@email.com",
  "password": "$2a$10$EnRvGzG7uJW6XaWvEumanu03qbUXdAanF8.96nPJTxq/O.EtwxMti",
  "role": "employer",
  "date": "2020-03-28T17:50:10.521Z",
  "requestSent": [],
  "requestRecieved": [],
  "__v": 0,
  "followed": [
    {
      "_id": "5e85bd74f9fd5c113013fb96",
      "requester": "5e7f8e4202a95532bce85947",
      "status": "pending",
      "updatedAt": "2020-04-02T10:24:52.289Z",
      "createdAt": "2020-04-02T10:24:52.289Z"
    }
    }
  ]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

Accept a request from a user:

Description:

A logged in User can Accept a request from another user,

- User must be logged in

UserA becomes a “follower” of UserB

```
// User B result

"_id": "5e9efe47263eb40fd0125b3b ",

"followed": [
    {
      "_id": "5e85bd74f9fd5c11",
      "follower": "5e7f8e4202a95532bce85947", // UserA _id
      "status": "connected",
      "updatedAt": "2020-04-02T10:24:52.289Z",
      "createdAt": "2020-04-02T10:24:52.289Z"
```

```
    }
```

UserA is “following” UserB

```
// User A (requester) result

"_id": "5e7f8e4202a95532bce85947",

"followed": [
    {
      "_id": "f9fd5c113013fb96",
      "following": "5e9efe47263eb40fd0125b3b", // UserB _id
      "status": "connected",
      "updatedAt": "2020-04-02T10:24:52.289Z",
      "createdAt": "2020-04-02T10:24:52.289Z"
```

URL

/users/addreQuest/:UserA/:UserB

Method

PUT

URL Params

:UserA = [string], :UserB = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
"_id": "5e7f8e5202a95532bce85948",
"name": "shane cunningham",
"email": "shane@email.com",
"password": "$2a$10$EnRvGzG7uJW6XaWvEumanu03qbUXdAanF8.96nPJTxq/O.EtwxMti",
"role": "employer",
"date": "2020-03-28T17:50:10.521Z",
"__v": 0,
"followed": [
  {
    "_id": "5e85bd74f9fd5c113013fb96",
    "requester": "5e7f8e4202a95532bce85947",
    "status": "pending",
```

```
      "updatedAt": "2020-04-02T10:24:52.289Z",
      "createdAt": "2020-04-02T10:24:52.289Z"
    }
    }
  ]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
Unauthorized
```


Remove following/ follower relationship:

Description:

A logged in User can Remove following/ follower relationship from the database,

- User must be logged in

URL

/users/remove/:UserA/:UserB

Method

PUT

URL Params

:UserA = [string], :UserB = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
  "_id": "5e9ee7fe40847127140ce90e",
  "name": "jack",
  "email": "jack@email.com",
  "password": "$2a$10$cp.LvWHZUGFeC2ZnCDRvI.Tn0Xe7yxSHrNtA8gbBeFqwXY6pCSbyK",
  "role": "employee",
  "date": "2020-04-21T12:33:02.933Z",
  "followed": [
    {
```

```
      "_id": "5e9f0394873c3213f8a7586b",
      "follower": "5e9ee81840847127140ce90f",
      "status": "connected",
      "createdAt": "2020-04-21T14:30:44.339Z"
    }
  ],
  "employment": [],
  "employeeDetails": [],
  "employeeCerts": [],
  "employerDetails": [],
  "insurance": [],
  "assets": [],
  "assetCerts": [],
  "safety_statements": [],
  "adminDetails": [],
  "intsituteDetails": [],
  "e_learning": [],
  "__v": 0
}
```


Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

models\follow.js

```
let followSchema = mongoose.Schema({
    requester: { type: mongoose.Schema.ObjectId, ref: 'User'},
    recipient: { type: mongoose.Schema.ObjectId, ref: 'User'},
    following: { type: mongoose.Schema.ObjectId, ref: 'User'},
    follower: { type: mongoose.Schema.ObjectId, ref: 'User'},
    status: {
      type: String,
      enums: ['requested', 'pending', 'connected']

    }
  }, {timestamps: true})

  module.exports = followSchema;
```

Follow Employer routes

Description:

The purpose of the follow Employer routes is to allow employee's to make connection requests to employers. When an employees' request is accepted by an employer they are kept on record. These routes are only used for read access only and allow for employee / employer (or admin/ institute) relationships.

Make a request to follow an Employer:

- User must be logged in

Employee id is pushed to Employer as "employees"

```
// Employer result

"_id": "5e9efe47263eb40fd0125b3b ",

"employment": [
    {
      "_id": "5e85bd74f9fd5c113013fb96",
      "employees": "5e7f8e4202a95532bce85947",
      "status": "requested",
      "updatedAt": "2020-04-02T10:24:52.289Z",
      "createdAt": "2020-04-02T10:24:52.289Z"
    }
```

Employee status shows that a request has been made to the Employer and the status is pending

```
// Employee result

"_id": "5e7f8e4202a95532bce85947",
"employment": [
    {
      "_id": "5e85bd74f9fd5c113013fb96",
      "employer": "5e9efe47263eb40fd0125b3b",
      "status": "pending",
      "updatedAt": "2020-04-02T10:24:52.289Z",
      "createdAt": "2020-04-02T10:24:52.289Z"
    }
```

URL

/users/followEmployer/:Employee/:Employer

Method

PUT

URL Params

:Employee = [string], :Employer = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  {
"_id": "5e9f71816a75f728b4998081",
"name": "HSE",
"email": "HSE@email.com",
"password": "$2a$10$t1XSqAGukFJnQhkp2KfhF..zEza6SqBxi.loRBnq9e9EdLob8CaQC",
"role": "institute",
"date": "2020-04-21T22:19:45.518Z",
"followed": [],
"employment": [],
"employeeDetails": [],
"employeeCerts": [],
"employerDetails": [],
"insurance": [],
"assets": [],
"assetCerts": [],
"safety_statements": [],
"adminDetails": [],
"instituteDetails": [],
"e_learning": [],
"__v": 0 } [
```


Error Response:

- **Code:** 500 Internal Server Error

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

Deny a request from a Employee:

Description:

A logged in Employer can reject a request from Employee, This action will pull the employment data created during the /users/followEmployer/:Employee/:Employer call from the database

- User must be logged in

URL

/users/deny/:Employee/:Employer

Method

PUT

URL Params

:Employee = [string], :Employer = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
"_id": "5e9f71816a75f728b4998081",
"name": "HSE",
"email": "HSE@email.com",
"password": "$2a$10$t1XSqAGukFJnQhkp2KfhF..zEza6SqBxi.loRBnq9e9EdLob8CaQC",
"role": "institute",
"date": "2020-04-21T22:19:45.518Z",
"followed": [],
"employment": [
  {
    "_id": "5ea5e957fbb5a26b7c10bbb5",
    "employees": "5e9f6ecd6a75f728b4998080",
    "status": "requested",
    "updatedAt": "2020-04-26T20:04:39.008Z",
    "createdAt": "2020-04-26T20:04:39.008Z"
  }
],
"employeeDetails": [],
"employeeCerts": [],
"employerDetails": [],
```

```
  "insurance": [],
  "assets": [],
  "assetCerts": [],
  "safety_statements": [],
  "adminDetails": [],
  "instituteDetails": [
    {
      "_id": "5e9f71ad6a75f728b4998082",
      "company_name": "HSE",
      "business_type": "Health care",
      "company_logo": "white-image.png",
      "business_email": "HSE@email.com",
      "phone": "123456123",
      "address": "address,address,address"
    }
  ],
  "e_learning": [],
  "__v": 0    }
  ]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
  "ok": 0,
  "errmsg": "E11000 duplicate key error collection: rcwdb.users index: name_1
dup key: { name: null }",
  "code": 11000,
 "codeName": "DuplicateKey",
  "keyPattern": {
    "name": 1
  },
 "keyValue": {
    "name": null
  },
  "name": "MongoError"
}
```

Accept a request from Employee:

Description:

A logged in Employer can Accept a request from Employee,

- User must be logged in

“employees” becomes a “my_employee” of Employer, status changes from “requested” to “connected”

```
// Employer result

"_id": "5e9efe47263eb40fd0125b3b ",

"employment": [
    {
      "_id": "5e85bd74f9fd5c11",
      "my_employee": "5e7f8e4202a95532bce85947", // Employee _id
      "status": "connected",
      "createdAt": "2020-04-02T10:24:52.289Z"
    }
```

Employee “employer” is changed to “my_employer”, status becomes “connected”

```
// Employee result

"_id": "5e7f8e4202a95532bce85947",

"employment": [
    {
      "_id": "f9fd5c113013fb96",
      "my_employer": "5e9efe47263eb40fd0125b3b", // Employer _id
      "status": "connected",
      "createdAt": "2020-04-02T10:24:52.289Z"
}
```

URL

/users/accept/:Employee/:Employer

Method

PUT

URL Params

:Employee = [string], :Employer = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
"_id": "5e9ee81840847127140ce90f",
"name": "austin",
"email": "austin@email.com",
"password": "$2a$10$gTN48uAtnAARibDx4nCFFe3iNWjcjI4AmsZm7FQnui/UF0WemmceS",
"role": "employer",
"date": "2020-04-21T12:33:28.340Z",
"followed": [],
"employment": [
  {
    "_id": "5e9ef87f52a12f2fe05af353",
    "employees": "5e9ee7fe40847127140ce90e",
    "status": "requested",
    "updatedAt": "2020-04-21T13:43:27.194Z",
    "createdAt": "2020-04-21T13:43:27.194Z"
  },
  {
    "_id": "5e9ef8d352a12f2fe05af357",
    "my_employee": "5e9ee7fe40847127140ce90e",
    "status": "connected",
    "createdAt": "2020-04-21T13:44:51.772Z"
  }
],
"employeeDetails": [],
"employeeCerts": [],
"employerDetails": [],
"insurance": [],
"assets": [],
"assetCerts": [],
"safety_statements": [],
"adminDetails": [],
"intsituteDetails": [],
"e_learning": [],      "__v": 0    }  ]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
  "ok": 0,
  "errmsg": "E11000 duplicate key error collection: rcwdb.users index: name_1
dup key: { name: null }",
  "code": 11000,
 "codeName": "DuplicateKey",
  "keyPattern": {
    "name": 1
  },
 "keyValue": {
    "name": null
  },
  "name": "MongoError"
}
```

Remove Employee/ Employer relationship:

Description:

A logged in User can Remove Employee/Employer relationship from the database,

- User must be logged in
- User must have a relevant id

URL

/users/drop/:UserA/:UserB

Method

PUT

URL Params

:UserA = [string], :UserB = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
  "_id": "5e9ee7fe40847127140ce90e",
  "name": "jack",
  "email": "jack@email.com",
  "password": "$2a$10$cp.LvWHZUGFeC2ZnCDRvI.Tn0Xe7yxSHrNtA8gbBeFqwXY6pCSbyK",
  "role": "employee",
  "date": "2020-04-21T12:33:02.933Z",
  "followed": [
    {
      "_id": "5e9f0394873c3213f8a7586b",
      "follower": "5e9ee81840847127140ce90f",
      "status": "connected",
      "createdAt": "2020-04-21T14:30:44.339Z"
    }
  ],
  "employment": [],
  "employeeDetails": [],
  "employeeCerts": [],
  "employerDetails": [],
  "insurance": [],
  "assets": [],
  "assetCerts": [],
  "safety_statements": [],
  "adminDetails": [],
  "intsituteDetails": [],
  "e_learning": [],
  "__v": 0
}
```


Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
  "stringValue": "\"5e9816a75f728b4998081\"",
  "kind": "ObjectId",
  "value": "5e9816a75f728b4998081",
  "path": "my_employer",
```

```
  "reason": {},
  "message": "Cast to ObjectId failed for value \"5e9816a75f728b4998081\" at
path \"my_employer\"",
  "name": "CastError"}
```

models\worker.js

```
let workersSchema = mongoose.Schema({
    employees:{type: mongoose.Schema.ObjectId, ref: 'User'},
    my_employee:{type: mongoose.Schema.ObjectId, ref: 'User'},

    employer:{type: mongoose.Schema.ObjectId, ref: 'User'},
    my_employer:{type: mongoose.Schema.ObjectId, ref: 'User'},

    status: {
        type: String,
        enums: ['requested', 'pending', 'connected']
    }
}, {timestamps: true})

module.exports = workersSchema;
```

Delete User Account

Description

Delete User Account by Id,

- User must be logged in,
- User has to have the same Id,

URL

/users/delete_account/:id

Method

GET

URL Params

Required:

id = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
'Account Deleted '
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "Unauthorized"
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

employers.js

models\employer.js

```
// employer details Schema
let employerSchema = mongoose.Schema({

    company_name:{
        type: String
    },
    business_type:{
        type: String,
        enum:[
            'Water',
            'Roads',
            'Environment',
            'Construction',
            'Quarying',
            'Administration',
            'Agricultural Services & Products',
            'Automotive',
            'Fishing',
            'Oil & Gas',
            'Pharmaceutical',
            'Retail and Hospitality',
            'Haulage & Transportation',
            'Security',
            'Healthcare',
            'Childcare',
            'Forestry',
            'Employee'
        ],
        required: true
    },
    company_logo:{
        type: String
    },
    business_email:{
        type: String
    },
    phone:{
        type: String
    },
    address:{
        type: String
    }
});
module.exports = employerSchema;
```

View all employers

Description:

A logged in User can view the list of employers

URL

/employers/viewall

Method

GET

URL Params

Required:

JWT (User must be logged in to view)

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
[
{
  "_id": "5e9ee81840847127140ce90f",
  "name": "austin",
  "employerDetails": [
    {
      "_id": "5e9f1a0bdd1daf17d8d89ac3",
      "company_name": "Construction Concrete",
      "business_type": "Construction",
      "company_logo": "QrTag.png",
      "business_email": "concon@email.com",
      "phone": "085-2835638 update",
      "address": "Abbeyside,Dungarvan,Co.Waterford"
    },
    {
      "_id": "5ea230b5f54ee4554c80946f",
      "business_type": "Construction",
      "company_logo": "QrTag.png",
      "business_email": "concon@email.com",
      "phone": "085-2835638",
      "address": "Dungarvan,Co.Waterford"
```

```
      }
    ]
  },
  {
    "_id": "5e9efe47263eb40fd0125b3b",
    "name": "shane",
    "employerDetails": [
      {
        "_id": "5e9f15d08725af1c08d2a229",
        "company_name": "Construction Concrete",
        "business_type": "Construction",
        "company_logo": "QrTag.png",
        "business_email": "concon@email.com",
        "phone": "085-2835638",
        "address": "Dungarvan,Co.Waterford"
      }
    ]
  }
]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
Unauthorized
```


View Employer by Id (isFollowing() employer)

Description
View Employer by Id,

- User must be logged in,
- User has to be following the employer,
- User must be **admin, employer, institute**

URL

/employers/view/:id (eg. ‘ /employers/view/5e68cd5507eb1121d80f9cc6 ‘)

Method:

GET

URL Params
Required:
id = [string], role = [‘admin’, ‘employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  "_id": "5e9ee81840847127140ce90f",
  "name": "austin",
  "email": "austin@email.com",
  "followed": [
    {
      "_id": "5e9f00d8263eb40fd0125b4b",
      "following": {
        "_id": "5e9efe59263eb40fd0125b3c",
        "name": "john",
        "role": "employee"
      },
      "status": "connected",
      "createdAt": "2020-04-21T14:19:04.390Z"
    },
    {
      "_id": "5e9f7a7e40a5691430e2583d",
      "follower": {
        "_id": "5e9f71816a75f728b4998081",
        "name": "HSE",
        "role": "institute"
      },
      "status": "connected",
      "createdAt": "2020-04-21T22:58:06.977Z"
    },
    {
      "_id": "5ea1af825d015451b85aa6b1",
      "follower": {
        "_id": "5e9f6ecd6a75f728b4998080",
        "name": "steve",
        "role": "admin"
      },
      "status": "connected",
      "createdAt": "2020-04-23T15:08:50.944Z"
    }
  ],
  "employment": [
    {
      "_id": "5e9f0abe64ed714e1015c7b3",
      "employees": {
        "_id": "5e9ee7fe40847127140ce90e",
        "name": "jack",
        "employeeDetails": [
```

```
          {
            "photo_id": "QrTag.png"
          }
        ]
      },
      "status": "requested",
      "updatedAt": "2020-04-21T15:01:18.252Z",
      "createdAt": "2020-04-21T15:01:18.252Z"
    },
    {
      "_id": "5ea1ada65d015451b85aa6ad",
      "my_employee": {
        "_id": "5e9f6ecd6a75f728b4998080",
        "name": "steve",
        "employeeDetails": []
      },
      "status": "connected",
      "createdAt": "2020-04-23T15:00:54.613Z"
    }
  ],
  "employerDetails": [
    {
      "_id": "5e9f1a0bdd1daf17d8d89ac3",
      "company_name": "Construction Concrete",
      "business_type": "Construction",
      "company_logo": "QrTag.png",
      "business_email": "concon@email.com",
      "phone": "085-2835638 update",
      "address": "Abbeyside,Dungarvan,Co.Waterford"
    }
  ]
}
```


Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

- **Code:** 401 Unauthorized

  Content:


```
{
       "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

View Own Employer Details by Id

Description
View Employer by Id,

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

URL

/employers/view_own/:id (eg. ‘ /employers/ view_own/5e68cd5507eb1121d80f9cc6 ‘)

Method:

GET

URL Params
Required:
id = [string], role = [‘employer’]
Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  "_id": "5e9ee81840847127140ce90f",
  "name": "austin",
  "email": "austin@email.com",
  "followed": [
    {
      "_id": "5e9f00d8263eb40fd0125b4b",
      "following": {
        "_id": "5e9efe59263eb40fd0125b3c",
        "name": "john",
        "role": "employee"
      },
      "status": "connected",
      "createdAt": "2020-04-21T14:19:04.390Z"
    },
    {
      "_id": "5e9f7a7e40a5691430e2583d",
      "follower": {
        "_id": "5e9f71816a75f728b4998081",
        "name": "HSE",
        "role": "institute"
      },
      "status": "connected",
      "createdAt": "2020-04-21T22:58:06.977Z"
    },
    {
      "_id": "5ea1af825d015451b85aa6b1",
      "follower": {
        "_id": "5e9f6ecd6a75f728b4998080",
        "name": "steve",
        "role": "admin"
      },
      "status": "connected",
      "createdAt": "2020-04-23T15:08:50.944Z"
    }
  ],
  "employment": [
    {
      "_id": "5e9f0abe64ed714e1015c7b3",
      "employees": {
        "_id": "5e9ee7fe40847127140ce90e",
        "name": "jack",
        "employeeDetails": [
```

```
          {
            "photo_id": "QrTag.png"
          }
        ]
      },
      "status": "requested",
      "updatedAt": "2020-04-21T15:01:18.252Z",
      "createdAt": "2020-04-21T15:01:18.252Z"
    },
    {
      "_id": "5ea1ada65d015451b85aa6ad",
      "my_employee": {
        "_id": "5e9f6ecd6a75f728b4998080",
        "name": "steve",
        "employeeDetails": []
      },
      "status": "connected",
      "createdAt": "2020-04-23T15:00:54.613Z"
    }
  ],
  "employerDetails": [
    {
      "_id": "5e9f1a0bdd1daf17d8d89ac3",
      "company_name": "Construction Concrete",
      "business_type": "Construction",
      "company_logo": "QrTag.png",
      "business_email": "concon@email.com",
      "phone": "085-2835638 update",
      "address": "Abbeyside,Dungarvan,Co.Waterford"
    }
  ]

}
```


Error Response:

- **Code:** 401 Unauthorized
  Content:

```
Unauthorized
```

- **Code:** 401 Unauthorized
  Content:


```
{
```

```
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

Delete employer details

Description

Delete Employer details by Id,

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

URL

/employers/delete/:id

Method

PUT

URL Params
Required:
id = [string], role = [‘employer’]
Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
      "n": 1,
      "nModified": 0,
      "ok": 1
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content"
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Add New Employer details

Description:

A logged in User can add new employer details { business_type, company_name, sampleFile, business_email, phone, address } 'sampleFile' is used for express fileupload for images and files.

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

Method:

PUT

URL

/employers/add/:id (must include the params id, this _id is the object Id created during registration)

URL Params

Required:

id = [string], role = ['employer']

Data Params

Json

Success Response:

- **Code:** 200

  Content:


```
{
"_id": "5e9ee81840847127140ce90f",
"employerDetails": [
  {
    "_id": "5e9f1a0bdd1daf17d8d89ac3",
    "company_name": "Construction Concrete",
    "business_type": "Construction",
    "company_logo": "QrTag.png",
    "business_email": "concon@email.com",
    "phone": "085-2835638 update",
    "address": "Abbeyside,Dungarvan,Co.Waterford"
  }
],
"__v": 1
}
```


Error Response

- **Code:** 401 Unauthorized

  Content:


```
{
       "error": "You are not authorized to view this content"
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
     "error": "No user found."
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
     "error": "No Employer Details found."
}
```

Edit Employer Details

Description:

A logged in User can edit existing employer details { business_type, company_name, business_email, phone, address }

Method:

PUT

URL

/employers/edit/:id/:docID (must include include entry :id and :docID params to pass)

URL Params

Required:

id = [string], docID = [string], role = ['employer']

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5e9f1a0bdd1daf17d8d89ac3",
  "company_name": "Construction Concrete",
  "business_type": "Construction",
  "company_logo": "QrTag.png",
  "business_email": "concon@email.com",
  "phone": "085-2835638 update",
  "address": "Abbeyside,Dungarvan,Co.Waterford"
}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."
}
```

Update Employer Company Logo

Description:

A logged in User can edit existing employer Company Logo. Use sampleFile instead of company company_logo on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

Method:
PUT
URL

/employers/updateLogo/:id/:docID (must include include entry :id and :docID params to pass)

URL Params

Required:
id = [string], docID = [string], role = ['employer']

Data Params

Json
Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5e9f1a0bdd1daf17d8d89ac3",
  "company_name": "Construction Concrete",
  "business_type": "Construction",
  "company_logo": "QrTag.png",
  "business_email": "concon@email.com",
  "phone": "085-2835638 update",
  "address": "Abbeyside,Dungarvan,Co.Waterford"          }
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."
}
```

insuredoc.js

models\insuredocs.js

```
//  Schema  for Insurance Documents
let insurSchema = mongoose.Schema({

    file_type:{
        type: String
    },
    file_name:{
        type: String
    },
    updated:{
        type: Date,
        default: Date.now
    },
    status: {
        type: String,
        enum: ['approved','pending','unapproved'],
        default: 'pending'
    },
    comment:{
        type: String
    },
    // Sets up the issue and expiry dates of the document, for poten-
tial front-end plugin
    expiry_date:{
        type: String

    },
    issue_date:{
        type: String
    }
});
module.exports = insurSchema;
```

View all Insurance docs

Description:

A User can view the list of insurance docs

- User must be logged in,
- User must be an **admin, employer, institute**
- User must be a follower of **employer**

```
   .where('role').equals('employer')
    .where('followed.follower').equals(user.id);
```

URL

/insuredoc/viewall

Method

GET

URL Params

Required:

id = [string], role = [‘admin’, ’employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
 [
     {
   "_id": "5e9ee81840847127140ce90f",
   "name": "austin",
   "employerDetails": [
     {
       "company_name": "Construction Concrete",
       "company_logo": "QrTag.png"
     }
   ],
   "insurance": []
 }
]
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

View by Id (isFollowing() employer)

Description

View insurance doc by Id, includes company_name, company_logo

- User must be logged in,
- User has to be following the employer,
- User must be **admin, employer, institute**

URL

/insuredoc/byEmployer/:id

Method:

GET

URL Params

Required:
id = [string], role = [**admin, employer, institute**]
Data Params

None
Success Response:

- **Code:** 200

  Content:


```
{
 "_id": "5e69495768ffe90100d32ad1",
  "company_name": "Construction Concrete",
  "company_logo": "logo.png",
  "insurance": [
    {
      "file_type": "file.pdf",
      "status": "invalid",
      "updated": "2020-03-13T13:37:21.162Z",
      "_id": "5e6b8c915f43320c7021c03f",
      "file_name": "old"
    },
    {
      "file_type": "file.png",
      "status": "pending",
      "updated": "2020-03-13T20:01:01.139Z",
      "_id": "5e6be67ddb586924c4501409",
      "file_name": "Insurance brand new"
    }    }
  ]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to view this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

View Own insuredocs by Id

Description
View Own insurance docs by Id

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

URL

/insuredoc/myDocs/:id

Method:

GET

URL Params

Required:
id = [string], role = ['employer']

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
"_id": "5e69495768ffe90100d32ad1",
  "company_name": "Construction Concrete",
  "company_logo": "logo.png",
  "insurance": [
    {
      "file_type": "file.pdf",
      "status": "invalid",
      "updated": "2020-03-13T13:37:21.162Z",
      "_id": "5e6b8c915f43320c7021c03f",
      "file_name": "old"
    },
```

```
    {
      "file_type": "file.png",
      "status": "pending",
      "updated": "2020-03-13T20:01:01.139Z",
      "_id": "5e6be67ddb586924c4501409",
      "file_name": "Insurance brand new"                } }]
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to write this content" }
```


View individual Insurance documents by Employer

Description

View Individual insuredoc by User Id and certificate Id

- User must be logged in,
- User has to be following the employer,
- User must be **admin, employer, institute**

URL

/insuredoc/:id/:docID

Method:

GET

URL Params

Required:

id = [string], id = [string], role = [**admin, employer, institute**]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Insurance brand Updated",
  "file_type": "QrTag.png",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"
}
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to view this content" }
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


View Own individual Insurance document by Employer

Description
View Individual data by User Id and certificate Id, User must be logged in,

- User has to have same id,
- The assets id must be included as a param
- User must be **employer**

URL

/insuredoc/view_own/:id/:docID

Method:

GET

URL Params

Required:
id = [string], docID = [string], role = [**employer**]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Insurance brand Updated",
  "file_type": "QrTag.png",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Add New Insurance Doc

Description:

A logged in User can add a new insuredoc {file_name, expiry_date, issue_date, sample-File} 'sampleFile' is used for express fileupload for images and files.

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

```
employer.insurance.push({
      file_name: req.body.file_name,
      file_type: file_type,
      issue_date: Date.parse(req.body.issue_date),
      expiry_date: Date.parse(req.body.expiry_date),
      createdBy: req.params.id
                });
```

issue_date and expiry_date use Date.parse() to convert the date string to a number value. The number value can be manipulated later in the front end.

issue_date and expiry_date should use an input value of type ="datetime-local"

```
<input name="" type="datetime-local" >
```

Date.parse info: https://www.w3schools.com/jsref/jsref_parse.asp

datetime-local info: https://www.w3schools.com/tags/att_input_type_datetime-local.asp

Method:
PUT

URL

/insuredoc/add/:id

URL Params

Required:
id = [string], role = ['employer']

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
"_id": "5e9ee81840847127140ce90f",
"insurance": [
  {
```

```
      "status": "pending",
      "_id": "5eadd1aeda5a0b1e0cba85fc",
      "file_name": "Insurance brand new",
      "file_type": "QrTag.png",
      "issue_date": "1553126400000",
      "expiry_date": "1584748800000",
      "updated": "2020-05-02T20:01:50.366Z"
    }
  ],
  "__v": 1 }
```


Error Response

- **Code:** 401 Unauthorized
  Content:


```
{
        "error": "You are not authorized to view this content"
}
```


- **Code:** 422 Unprocessable Entity
  Content:


```
{
      "error": "No user found."
}
```


- **Code:** 422 Unprocessable Entity
  Content:


```
{
      "error": "No Employer Details found."
}
```


Edit Insurance Document Details

Description:
A logged in User can edit existing insuredoc details

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

This route allows for one or all of the following fields to be updated:

- file_name
- issue_date
- expiry_date

Method:

PUT

URL

/insuredoc/editDoc/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employer’]

Data Params

Json

Success Response:

- **Code:** 200
  Content:


```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Insurance brand Updated",
  "file_type": "QrTag.png",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"
}
```


Error Response

- **Code:** 401 Unauthorized
  Content:


```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."
}
```

Update Insuredoc file_type

Description:

A logged in User can update existing insurance document file_type. Use sampleFile instead of company file_type on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

Method:

PUT

URL

/insuredoc/updateFile/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employer’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Insurance brand Updated",
  "file_type": "QrTag.png",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"            }
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
       "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."
}
```

Delete insuredoc

Description

Delete insuredoc by Id,

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

URL

/insuredoc/delete/:id

Method

PUT

URL Params
Required:
id = [string], role = [‘employer’]
Data Params

None

Success Response:

- **Code:** 200
  Content:

```
{
      "n": 1,
      "nModified": 0,
      "ok": 1
}
```

Error Response:

- **Code:** 401 Unauthorized
  Content:

```
{
  "error": "You are not authorized to view this content"
}
```

- **Code:** 500 Internal Server Error
  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Review insuredoc

Description:

The purpose of this route is to allow a user that is a follower of the route :id write access to certain fields. The user can verify the document by changing the **status** field to one of the

following enum values [**‘approved’, ‘pending’, ‘unapproved’**] or the user may alter the **comment** field. For a User to edit an existing insuredoc

- User must be logged in,
- User has to be a follower of the Id,
- User must be an **admin, employer, institute**

Method:

PUT

URL

/insuredoc/editDoc/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘admin’, ‘employer’, ‘institute’]

Data Params

Json

Success Response:

- **Code:** 200
  Content:


```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Insurance brand Updated",
  "file_type": "QrTag.png",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"
}
```


Error Response

- **Code:** 401 Unauthorized
  Content:


```
{
```

```
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."       }
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

safedocs.js

models\safedoc.js

```
//  Schema  for Safety Statements
let safeSchema = mongoose.Schema({

    file_type:{
        type: String
    },
    file_name:{
        type: String
    },
    updated:{
        type: Date,
        default: Date.now
    },
    status: {
        type: String,
        enum: ['approved','pending','unapproved'],
        default: 'pending'
    },
    comment:{
        type: String
    },
    // Sets up the issue and expiry dates of the document, for poten-
tial front-end plugin
    expiry_date:{
        type: String
    },
    issue_date:{
        type: String
    }
});
```

View all Safety docs

Description:

A User can view the list of safety docs

- User must be logged in,
- User must be an **admin, employer, institute**
- User must be a follower of **employer**

```
})
   .where('role').equals('employer')
    .where('followed.follower').equals(user.id)
   ;
```

URL

/safedocs/viewall

Method

GET

URL Params

Required:

id = [string], role = [‘admin’, ’employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
[
    {
  "_id": "5e9ee81840847127140ce90f",
  "name": "austin",
  "employerDetails": [
    {
      "company_name": "Construction Concrete",
      "company_logo": "QrTag.png"
    }
  ],
  "safety_statements": [
      {
    "status": "pending",
    "updated": "2020-04-24T12:13:30.539Z",
    "_id": "5ea2d7ea3e85836014b9d954",
    "file_name": "safety 4",
    "file_type": "QrTag.png"
  },
  {
    "status": "pending",
    "updated": "2020-04-24T12:15:12.295Z",
    "_id": "5ea2d8503e85836014b9d956",
    "file_name": "safety 4",
    "file_type": "QrTag.png"
  }
    ]
 }
]
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

View by Id (isFollowing() employer)

Description
View safety doc by Id, includes company_name, company_logo

- User must be logged in,
- User has to be following the employer,
- User must be **admin, employer, institute**

URL

/safedocs/byEmployer/:id

Method:

GET

URL Params

Required:
id = [string], role = [**admin, employer, institute**]
Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  "_id": "5e9ee81840847127140ce90f",
  "name": "austin",
  "employerDetails": [
    {
      "company_name": "Construction Concrete",
      "company_logo": "QrTag.png"
    }
  ],
  "safety_statements": [
        {
      "status": "pending",
      "updated": "2020-04-24T12:13:30.539Z",
```

```
      "_id": "5ea2d7ea3e85836014b9d954",
      "file_name": "safety 4",
      "file_type": "QrTag.png"
    },
    {
      "status": "pending",
      "updated": "2020-04-24T12:15:12.295Z",
      "_id": "5ea2d8503e85836014b9d956",
      "file_name": "safety 4",
      "file_type": "QrTag.png"
    }
]
}
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to view this content" }
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


View Own safedocs by Id

Description

View Own safety docs by Id

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

URL

/safedocs/myDocs/:id

Method:

GET

URL Params

Required:
id = [string], role = [‘employer’]

Data Params

None

Success Response:

- **Code:** 200
  Content:


```
{
"_id": "5e9ee81840847127140ce90f",
  "name": "austin",
  "employerDetails": [
    {
      "company_name": "Construction Concrete",
      "company_logo": "QrTag.png"

  ],
  "safety_statements": [
    {
      "status": "pending",
      "updated": "2020-04-24T12:13:30.539Z",
      "_id": "5ea2d7ea3e85836014b9d954",
      "file_name": "Update 15:47",
      "file_type": "white-image.png"
    },
    {
      "status": "pending",
      "updated": "2020-04-24T12:15:12.295Z",
      "_id": "5ea2d8503e85836014b9d956",
      "file_name": "safety 4",
      "file_type": "QrTag.png"
    }}]
```


Error Response:

- **Code:** 500 Internal Server Error
  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to write this content" }
```

View individual Safety statements by Employer

Description
View Individual safedocs by User Id and certificate Id

- User must be logged in,
- User has to be following the employer,
- User must be **admin, employer, institute**

URL

/safedocs/:id/:docID

Method:

GET

URL Params

Required:
id = [string], id = [string], role = [**admin, employer, institute**]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Safe",
  "file_type": "QrTag.png",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"
}
```

Error Response:

- **Code:** 401 Unauthorized

Content:

```
{
  "error": "You are not authorized to view this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

View Own individual Safety document by Employer

Description
View Individual data by User Id and document Id, User must be logged in,

- User has to have same id,
- The assets id must be included as a param
- User must be **employer**

URL

/safedocs/view_own/:id/:docID

Method:

GET

URL Params

Required:
id = [string], docID = [string], role = [**employer**]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
   "status": "pending",
  "updated": "2020-04-24T11:57:20.902Z",
  "_id": "5ea2d4204078070ac48283c6",
  "file_name": "safety 4",
  "file_type": "file.png"
```

```
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Add New Safety Statement

Description:

A logged in User can add a new safedocs {file_name, expiry_date, issue_date, sample-File} 'sampleFile' is used for express fileupload for images and files.

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

```
employer. safety_statements.push({
      file_name: req.body.file_name,
      file_type: file_type,
      issue_date: Date.parse(req.body.issue_date),
      expiry_date: Date.parse(req.body.expiry_date),
      createdBy: req.params.id
                });
```

issue_date and expiry_date use Date.parse() to convert the date string to a number value. The number value can be manipulated later in the front end.

issue_date and expiry_date should use an input value of type =”datetime-local”

```
<input name="" type="datetime-local" >
```

Date.parse info: https://www.w3schools.com/jsref/jsref_parse.asp

datetime-local info: https://www.w3schools.com/tags/att_input_type_datetime-local.asp

Method:

PUT

URL

/safedocs/add/:id

URL Params

Required:
id = [string], role = [‘employer’]

Data Params

Json
Success Response:

- **Code:** 200
  Content:


```
{
"_id": "5e9ee81840847127140ce90f",
"safety_statements": [
  {
    "status": "pending",
    "updated": "2020-04-24T12:13:30.539Z",
    "_id": "5ea2d7ea3e85836014b9d954",
    "file_name": "safety 4",
    "file_type": "QrTag.png"
  }  ]
"__v": 1 }
```


Error Response

- **Code:** 401 Unauthorized
  Content:


```
{
       "error": "You are not authorized to view this content"
}
```


- **Code:** 422 Unprocessable Entity
  Content:


```
{
      "error": "No user found."
}
```


- **Code:** 422 Unprocessable Entity
  Content:


```
{
      "error": "No Employer Details found."}
```

Edit Safety statement Details

Description:

A logged in User can edit existing safedocs details, excluding **file_type (sampleFile)** field

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

This route allows for one or all of the following fields to be updated:

- file_name
- issue_date
- expiry_date

Method:

PUT

URL

/safedocs/edit/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = ['employer']

Data Params

Json

Success Response:

- **Code:** 200
  Content:


```
{
  "status": "pending",
  "updated": "2020-04-24T12:13:30.539Z",
  "_id": "5ea2d7ea3e85836014b9d954",
  "file_name": "Update 13:02",
  "file_type": "white-image.png",
  "issue_date": "March 21, 2020",
  "expiry_date": "March 21, 2021"}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Certificate Details found."
}
```

Update safedocs file_type

Description:

A logged in User can update existing safedocs file_type. Use sampleFile instead of company file_type on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

Method:

PUT

URL

/safedocs/updateFile/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employer’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Safe Updated",
  "file_type": "file.pdf",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"             }
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Certificate Details found."
}
```

Delete safedocs

Description

Delete safedocs by Id,

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

URL

/safedocs/delete/:id

Method

PUT

URL Params
Required:
id = [string], role = [‘employer’]
Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
      "n": 1,
      "nModified": 0,
      "ok": 1
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content"
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Review safedocs

Description:

The purpose of this route is to allow a user that is a follower of the route :id write access to certain fields. The user can verify the document by changing the **status** field to one of the following enum values [**'approved', 'pending', 'unapproved'**] or the user may alter the **comment** field. For a User to edit an existing safety statement

- User must be logged in,
- User has to be a follower of the Id,
- User must be an **admin, employer, institute**

Method:

PUT

URL

/safedocs/review/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = ['admin', 'employer', 'institute']

Data Params

Json

Success Response:

- **Code:** 200
  Content:

```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "safe brand Updated",
  "file_type": "safety.pdf",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "This is where my feedback goes as a String value"
}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."       }
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

asset.js

models\assets.js

```
// assets Schema
let assetSchema = mongoose.Schema({
    asset_type:{
        type: String,
        enum:[
            'Water',
            'Roads',
            'Environment',
            'Construction',
            'Quarying',
            'Administration',
            'Agricultural Services & Products',
            'Automotive',
            'Fishing',
            'Oil & Gas',
            'Pharmaceutical',
            'Retail and Hospitality',
            'Haulage & Transportation',
            'Security',
            'Healthcare',
            'Childcare',
            'Forestry',
            'Employee'
        ],
        required: true
    },
    status:{
        type:String,
        enum:['Active','Inactive'],
        default:'Inactive'
    },
    asset_image:{
        type: String
    },
    QR_Tag:{
        type: String
    },
    asset_name:{
        type: String
    },
    firstname:{
        type: String
    },
```

```
    lastname:{
        type: String
    },
    mobile_phone:{
        type: String
    },
    photo_id:{
        type: String
    },
    email:{
        type: String
    }
});
```

View all assets

Description:

A User can view the list of assets

- User must be logged in,
- User must be an **admin, employer, institute**
- User must be a follower of **employer**

```
})
   .where('role').equals('employer')
    .where('followed.follower').equals(user.id)
   ;
```

URL

/asset/viewall

Method

GET

URL Params

Required:

id = [string], role = [‘admin’, ’employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
[
  {
  "_id": "5e9ee81840847127140ce90f",
  "name": "austin",
  "employerDetails": [
    {
      "company_name": "Construction Concrete",
      "company_logo": "QrTag.png"
    }
  ],
  "assets": [
    {
      "status": "Active",
      "_id": "5ea36437916c505e70e673bd",
      "asset_type": "Automotive",
      "asset_name": "Lorry"
    },
    {
      "status": "Inactive",
      "_id": "5ea42c7c236a0b572cfaf068",
      "asset_type": "Automotive",
      "asset_name": "Lorry"
    },
    {
      "status": "Active",
      "_id": "5ea8bd9477e7396b84ab934d",
      "asset_type": "Employee",
      "firstname": "Larry",
      "lastname": "Dunne",
      "mobile_phone": "123456781",
      "employee_email": "larry@email.com",
      "photo_id": "white-image.png"
    },
    {
      "status": "Active",
      "_id": "5ea8c6736a4e6031a0d72ede",
      "asset_type": "Employee",
      "firstname": "Billy",
      "lastname": "kenny",
      "mobile_phone": "123456781",
      "employee_email": "kenny@email.com",
      "photo_id": "white-image.png"
    }
```

```
    ]
  }]
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

View Asset Details by Id (isFollowing() employer)

Description
View Asset by Id, Includes asset certs

- User must be logged in,
- User has to be following the employer,
- User must be **admin, employer, institute**

URL

/asset/byEmployer/:id

Method:

GET

URL Params

Required:
id = [string], role = [**admin, employer, institute**]
Data Params

None
Success Response:

- **Code:** 200

  Content:

```
{
 "_id": "5e9ee81840847127140ce90f",
  "employerDetails": [
    {
      "company_name": "Construction Concrete",
      "company_logo": "QrTag.png"
    }
  ],

  "assets": [
    {
```

```
      "status": "Inactive",
      "_id": "5ea42e061d03ca3688d00fa2",
      "asset_type": "Automotive",
      "asset_name": "Lorry"
    },
    {
      "status": "Active",
      "_id": "5ea8bd9477e7396b84ab934d",
      "asset_type": "Employee",
      "firstname": "Larry",
      "lastname": "Dunne",
      "mobile_phone": "123456781",
      "email": "larry@email.com",
      "photo_id": "white-image.png"
    },
    {
      "status": "Active",
      "_id": "5ea8c6736a4e6031a0d72ede",
      "asset_type": "Employee",
      "firstname": "Billy",
      "lastname": "kenny",
      "mobile_phone": "123456781",
      "email": "kenny@email.com",
      "photo_id": "white-image.png"

  ],
  "assetCerts": [
    {
      "status": "pending",
      "comments": [],
      "updated": "2020-04-25T17:09:45.968Z",
      "shared": [
        {
          "_id": "5eacaefc14432f33a47e44f5",
          "requester": "5e9f6ecd6a75f728b4998080",
          "status": "requested"
        }
      ],
      "_id": "5ea46ed99a8310582432dcb5",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    },
    {
      "status": "pending",
      "comments": [],
```

```
      "updated": "2020-04-25T17:09:57.398Z",
      "_id": "5ea46ee59a8310582432dcb6",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    }
  ]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to write this content" }
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


View Own Asset Details by Id

Description
View Own Asset details by Id, Includes asset certs

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

URL

/asset/myAsset/:id

Method:

GET

URL Params

Required:
id = [string], role = [‘employer’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  "_id": "5e9ee81840847127140ce90f",
  "employerDetails": [
    {
      "company_name": "Construction Concrete",
      "company_logo": "QrTag.png"
    }
  ],
  "assets": [

    {
      "status": "Inactive",
      "_id": "5ea42e061d03ca3688d00fa2",
      "asset_type": "Automotive",
      "asset_name": "Lorry"
    },
    {
      "status": "Active",
      "_id": "5ea8bd9477e7396b84ab934d",
      "asset_type": "Employee",
      "firstname": "Larry",
      "lastname": "Dunne",
      "mobile_phone": "123456781",
      "employee_email": "larry@email.com",
      "photo_id": "white-image.png"
    },
    {
      "status": "Active",
      "_id": "5ea8c6736a4e6031a0d72ede",
      "asset_type": "Employee",
      "firstname": "Billy",
      "lastname": "kenny",
      "mobile_phone": "123456781",
      "employee_email": "kenny@email.com",
      "photo_id": "white-image.png"
    }
  ],
  "assetCerts": [
    {
      "status": "pending",
      "comments": [],
      "updated": "2020-04-25T16:48:46.135Z",
      "_id": "5ea469ee11332e4b8cb9bc38",
```

```
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    },
    {
      "status": "pending",
      "comments": [],
      "updated": "2020-04-25T17:18:01.384Z",
      "_id": "5ea470c9aa6fc935cc672d2b",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    }
  ]   },
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to write this content" }
```


View individual Asset Created by Employer

Description
View Individual Asset by User Id and Asset Id, Includes related certificates data

- User must be logged in,
- User has to be following the employer,
- User must be **admin, employer, institute**

URL

/asset/viewAsset/:id/:docID

Method:

GET

URL Params

Required:

id = [string], id = [string], role = [**admin, employer, institute**]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
 {
  "data": {
    "asset": {
      "status": "Inactive",
      "_id": "5ea35e7501d77208c09aff4f",
      "asset_type": "Automotive",
      "asset_name": "Lorry"
    },
    "certificates": []
  }
}
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to view this content" }
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

View Own individual Asset

Description
View Individual Asset by User Id and Asset Id, Includes related certificates data

- User must be logged in,
- User has to have same id,
- The assets id must be included as a param
- User must be **employer**

URL

/asset/viewMyAsset/:id/:docID

Method:

GET

URL Params

Required:
id = [string], id = [string], role = [**employer**]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
 {
  "data": {
   "asset": {
     "status": "Inactive",
     "_id": "5ea35e7501d77208c09aff4f",
     "asset_type": "Automotive",
     "asset_name": "Lorry"
   },
   "certificates": []
  }
}
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to view this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

View Own Employee Assets Created by Employer

Description
View Own employee Asset details by Id, Also includes followed employees not created by the employer

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

URL

/asset/view_my_employees/:id

Method:

GET

URL Params

Required:
id = [string], role = ['employer']

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
    "employee": [
    {
      "status": "Active",
      "_id": "5ea8bd9477e7396b84ab934d",
      "asset_type": "Employee",
      "firstname": "Larry",
      "lastname": "Dunne",
      "mobile_phone": "123456781",
```

```
      "email": "larry@email.com",
      "photo_id": "white-image.png"
    },
    {
      "status": "Active",
      "_id": "5ea8c6736a4e6031a0d72ede",
      "asset_type": "Employee",
      "firstname": "Billy",
      "lastname": "kenny",
      "mobile_phone": "123456781",
      "email": "kenny@email.com",
      "photo_id": "white-image.png"
    },
    {
      "status": "Active",
      "_id": "5eab4ea852b42f5448bfa9ef",
      "asset_type": "Employee",
      "firstname": "james",
      "lastname": "brown",
      "mobile_phone": "123456781",
      "email": "brown@email.com"
    },
    {
      "status": "Active",
      "_id": "5eab4f8b16085b2d046d2c3d",
      "asset_type": "Employee",
      "firstname": "mike",
      "lastname": "byrne",
      "mobile_phone": "123456781",
      "email": "mike@email.com"
    }
  ],
  "result": [
    {
      "following": {
        "_id": "5e9efe59263eb40fd0125b3c",
        "email": "john@email.com",
        "employeeDetails": [
          {
            "status": "Inactive",
            "_id": "5e9f469d2d7f2955b89caf6b",
            "firstname": "John",
            "lastname": "Cunningham",
            "mobile_phone": "4567345",
            "photo_id": "QrTag.png"
          }
        ]
      }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to write this content" }
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

View Employee Assets Created by Employer (isFollowing() employer)

Description
View employee Asset details by Id, Also includes followed employees not created by the employer

- User must be logged in,
- User must be a follower of the employer,
- User must be an **employer, admin, institute**

URL

/asset/view_ employees/:id

Method:

GET

URL Params

Required:
id = [string], role = [‘admin’, ‘employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
    "employee": [
    {
      "status": "Active",
      "_id": "5ea8bd9477e7396b84ab934d",
      "asset_type": "Employee",
      "firstname": "Larry",
      "lastname": "Dunne",
      "mobile_phone": "123456781",
      "email": "larry@email.com",
      "photo_id": "white-image.png"
    },
    {
      "status": "Active",
      "_id": "5ea8c6736a4e6031a0d72ede",
      "asset_type": "Employee",
      "firstname": "Billy",
      "lastname": "kenny",
      "mobile_phone": "123456781",
      "email": "kenny@email.com",
      "photo_id": "white-image.png"
    },
    {
      "status": "Active",
      "_id": "5eab4ea852b42f5448bfa9ef",
      "asset_type": "Employee",
      "firstname": "james",
      "lastname": "brown",
      "mobile_phone": "123456781",
      "email": "brown@email.com"
    },
    {
      "status": "Active",
      "_id": "5eab4f8b16085b2d046d2c3d",
      "asset_type": "Employee",
      "firstname": "mike",
      "lastname": "byrne",
      "mobile_phone": "123456781",
      "email": "mike@email.com"
    }
  ],
  "result": [
    {
```

```
      "following": {
        "_id": "5e9efe59263eb40fd0125b3c",
        "email": "john@email.com",
        "employeeDetails": [
          {
            "status": "Inactive",
            "_id": "5e9f469d2d7f2955b89caf6b",
            "firstname": "John",
            "lastname": "Cunningham",
            "mobile_phone": "4567345",
            "photo_id": "QrTag.png"
          }
        ]
      }
    },
    {},
    {}
  ]
}
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to write this content" }
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No user found."
}
```

View Own Equipment/Other Assets Created by Employer

Description
View Asset details by Id excluding employees

- User must be logged in,
- User have the same Id,
- User must be an **employer**

URL

/asset/view_my_equipment/:id

Method:

GET

URL Params

Required:
id = [string], role = ['employer']

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
    {
    "status": "Active",
    "_id": "5ea36437916c505e70e673bd",
    "asset_type": "Automotive",
    "asset_name": "Lorry"
  },
  {
    "status": "Inactive",
    "_id": "5ea42c7c236a0b572cfaf068",
    "asset_type": "Automotive",
    "asset_name": "van"
  }
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"}
```

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to write this content" }
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

View Equipment/Other Assets Created by Employer (isFollowing() employer)

Description
View Asset details by user Id, excluding employee

- User must be logged in,
- User must be a follower of the employer,
- User must be an **employer, admin, institute**

URL

/asset/view_ equipment/:id

Method:

GET

URL Params

Required:
id = [string], role = [‘admin’, ‘employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{{{
    "status": "Active",
    "_id": "5ea36437916c505e70e673bd",
    "asset_type": "Automotive",
    "asset_name": "Lorry"
  },
  {
    "status": "Inactive",
    "_id": "5ea42c7c236a0b572cfaf068",
    "asset_type": "Automotive",
    "asset_name": "van"
  }
}
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to write this content" }
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

Delete Asset Details by Id

Description

Delete Asset details by user Id and asset Id,

- User must be logged in,
- User has to have the same user Id,
- User must be **employer**

URL

/asset/delete/:id/:docID

Method

PUT

URL Params
Required:
id = [string], docID = [string], role = [employer]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
      "n": 1,
      "nModified": 0,
      "ok": 1
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content"
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Add New Asset equipment Details

Description:

A logged in User can add new asset details

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

Method:

PUT

URL

/asset/add/:id

URL Params

Required:
id = [string], role = [‘employer’]
Data Params

Json
Success Response:

- **Code:** 200

  Content:

```
{
    {
      "status": "Inactive",
      "_id": "5ea42e061d03ca3688d00fa2",
      "asset_type": "Automotive",
      "asset_name": "Lorry"
    }}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{ "error": "No Employee Details found." }
```

Add New Asset employee Details

Description:

A logged in User can add new asset employee details

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

Method:
PUT

URL

/asset/add_employee_asset/:id

URL Params

Required:
id = [string], role = [‘employer’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:


```
{
"_id": "5e9ee81840847127140ce90f",
  "assets": [
    {
      "status": "Active",
      "_id": "5eab4f8b16085b2d046d2c3d",
      "asset_type": "Employee",
      "firstname": "mike",
      "lastname": "byrne",
      "mobile_phone": "123456781",
      "email": "mike@email.com"
      }
    ]
```


Error Response

- **Code:** 401 Unauthorized

  Content:


```
{
        "error": "You are not authorized to view this content"
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{ "error": "No asset Details found." }
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Edit Asset Details by ID

Description:

A logged in User can edit existing asset details not including express fileupload sampleFile

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

Method:
PUT
URL

/asset/update/:id/:docID

URL Params

Required:
id = [string], docID = [string], role = [‘admin’]

Data Params

Json
Success Response:

- **Code:** 200

  Content:

```
{
  "status": "Active",
  "_id": "5ea36437916c505e70e673bd",
  "asset_type": "Automotive",
  "asset_name": "Lorry"
}
```

Success Response:

- **Code:** 200

  Content:

```
{
      "status": "Active",
      "_id": "5eab4ea852b42f5448bfa9ef",
      "asset_type": "Employee",
      "firstname": "james",
      "lastname": "brown",
      "mobile_phone": "123456781",
      "email": "brown@email.com"
    }
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Admin Details found."
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
"error": "Error reading user: CastError: Cast to ObjectId failed for value
\"5e9efe593eb40fd0125b3c\" at path \"_id\" for model \"User\""
}
```

Update Profile Image

Description:

A logged in User can edit existing employee Photo ID. Use sampleFile instead of photo_id on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

Requirements:

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

Method:

PUT

URL

/asset/add_profile_img/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = ['employer']

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
   "status": "Active",
  "_id": "5ea8c6736a4e6031a0d72ede",
  "asset_type": "Employee",
  "firstname": "Billy",
  "lastname": "kenny",
  "mobile_phone": "123456781",
  "employee_email": "kenny@email.com",
  "photo_id": "white-image.png"             }
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
```

```
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Asset Details found."
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
"error": "Error reading user: CastError: Cast to ObjectId failed for value
\"5e9efe593eb40fd0125b3c\" at path \"_id\" for model \"User\""
}
```

Add asset equipment image

Description:

A logged in User can edit existing asset image for a piece of equipment. Use sampleFile instead of asset_image on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

Requirements:

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

Method:

PUT

URL

/asset/addImage/:id/:docID

URL Params

Required:
id = [string], docID = [string], role = [‘employer’]

Data Params

Json
Success Response:

- **Code:** 200

  Content:


```
{
  "status": "Inactive",
  "_id": "5ea35e7501d77208c09aff4f",
  "asset_type": "Automotive",
  "asset_name": "Lorry",
  "asset_image": "white-image.png"
}
```


Error Response

- **Code:** 401 Unauthorized

  Content:


```
{     "error": "You are not authorized to view this content"        }
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{     "error": "No user found."      }
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No Admin Details found."
}
```


- **Code:** 500 Internal Server Error

  Content:


```
{
"error": "Error reading user: CastError: Cast to ObjectId failed for value
\"5e9efe593eb40fd0125b3c\" at path \"_id\" for model \"User\""
}
```

QR Tag

Description:

A logged in User can add or edit an existing QR_Tag. Use sampleFile instead of QR_Tag on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

Requirements:

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

Method:
PUT
URL

/asset/add_QR_Tag/:id/:docID

URL Params

Required:
id = [string], docID = [string], role = [‘employer’]

Data Params

Json
Success Response:

- **Code:** 200

  Content:


```
{

  "_id": "5e9ee81840847127140ce90f",
  "assets": [
    {
      "status": "Inactive",
      "_id": "5ea35e7501d77208c09aff4f",
      "asset_type": "Automotive",
      "asset_name": "Lorry",
      "asset_image": "white-image.png",
      "QR_Tag": "QrTag.png"
    }
      ]           }
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Asset Details found."
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
"error": "Error reading user: CastError: Cast to ObjectId failed for value
\"5e9efe593eb40fd0125b3c\" at path \"_id\" for model \"User\""
}
```

certificates.js

models\certificate.js

```
// certificates Schema
let certSchema = mongoose.Schema({
    assetID:{
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Asset'
    },

    water_section:{
        type: String,
        enum:[
          'Risk Assessment for Cryptosporidium',
          'Filters Evaluation Operation Maintenance',
```

```
                'Water Treatment - Dealing with Problems',
                'Drinking Water Regulations',
                'Drinking Water Incident Management Training',
                'Sustainable Urban Drainage System Training (SUDS)',
                'EPANET training',
                'Pumps Operation and Maintenance',
                'Water Clarification Process/THM Removal',
                'Drinking Water Regulations EPA Handbook',
                'Fluoridation of Water Supplies',
'WasteWater Treatment - Plant Operators Assessment Course including Practical assessment',
       'Water Treatment - Plant Operators Assessment Course including Practical assessment',
                'Distribution System Operations and Maintenance',
                'Water Treatment & Distribution - Appreciation',
                'Fats Oil and Grease',
                'Quality Assurance Water Treatment Plants',
                'Quality Assurance Wastewater Treatment Plants',
                'Chlorine Handling',
                'DWNMP Sampling Procedures',
                'Membrane Technology in Water / Wastewater Plants',
                'Operation and Maintenance of Small Wastewater Plants',
                'Safety for Water/Wastewater Workers',
                'Sludge Handling',
                'Taste and Odour Issues in Water Treatment',
                'Troubleshooting the Activated Sludge Process',
                'Water Conservation -Network Management - Leakage Control - Operatives',
                'Water Conservation - Network Management Leakage Control - Managers',
                'Pipeline Corrosion/Lead Services',
                'Confined Spaces - High Risk',
                'Confined Spaces - Low Risk',
                'Confined Spaces - Medium Risk',
                'Confined Spaces - Management of Risk',
                'Confined Spaces - Emergency & Rescue',
                'Distribution System - Unidirectional Flushing',
                'Water Metering and Installation',
                'Hygiene in Water Services',
                'Inspection of Domestic Waste Water Treatment Systems',
                'Management and Operation of Waste Water Overflows',
                'Nutrient Removal in Wastewater Treatment Plants',
                'Control Septicity in Rising Mains & Wastewater networks',
                'Electrofusion'
            ]

        },

        roads_section:{ type:String,
                enum:[
                    'CSCS - Mini Digger New Entrant Programme',
                    'CSCS - Health & Safety at Roadworks',
```

```
                    'CSCS - Tractor Dozer New Entrant Programme',
                    'CSCS - Locating Underground Services',
                    'Trench Support',
                    'Driver CPC Module 1 (CVEDT)',
                    'Driver CPC Module 2 (MRMET)',
                    'Driver CPC Module 3 (HSOPD)',
                    'Driver CPC Module 4 (RPDTI)',
                    'Driver CPC Module 5 (PROTD)',
                    'Driver CPC Module 6 (PROBD)',
                    'Manual Handling Outdoor ',
                    'CSCS - 360 Degree Excavator New Entrant Programme',
                    'IOSH - Managing Safely for Construction Managers ',
                    'Ride on Mower',
                    'Operation & Maintenance of Lawnmowers',
                    'CSCS - Slinger Signaller New Entrant Programme',
                    'CSCS - Signing Lighting & Guarding at Roadworks',
                    'Operation and Maintenance of Strimmers & Bushcutters',
                    'Safe Systems of Work Plans Training',
                    'CSCS - 180 Degree Excavator New Entrant Programme',
                    'CSCS - Site Dumper New Entrant Programme',
                'CSCS - 180 Degree Excavator Experienced Operator Programme / Assessment',
                    'Safe Pass',
                    'Basic Road Strengthening',
                    'Chainsaw Training For Local Authority Operatives - Refresher',
                    'Masonry Arch Repair',
                 'CSCS - 360 Degree Excavator Experienced Operator Programme / Assessment',
                    'CSCS - Mini Digger Experienced Operator Programme / Assessment',
                    'CSCS - Mini Digger New Entrant Programme',
                    'CSCS - Health & Safety at Roadworks',
                    'CSCS - Tractor Dozer New Entrant Programme',
                    'CSCS - Locating Underground Services',
                    'Trench Support',
                    'Driver CPC Module 1 (CVEDT)',
                    'Driver CPC Module 2 (MRMET)',
                    'Driver CPC Module 3 (HSOPD)',
                    'Driver CPC Module 4 (RPDTI)',
                    'Driver CPC Module 5 (PROTD)',
                    'Driver CPC Module 6 (PROBD)',
                    'Manual Handling Outdoor ',
                    'CSCS - 360 Degree Excavator New Entrant Programme',
                    'IOSH - Managing Safely for Construction Managers ',
                    'Ride on Mower',
                    'Operation & Maintenance of Lawnmowers',
                    'CSCS - Slinger Signaller New Entrant Programme',
                    'CSCS - Signing Lighting & Guarding at Roadworks',
                    'Operation and Maintenance of Strimmers & Bushcutters',
                    'Safe Systems of Work Plans Training',
                    'CSCS - 180 Degree Excavator New Entrant Programme',
```

```
                    'CSCS - Site Dumper New Entrant Programme',
                'CSCS - 180 Degree Excavator Experienced Operator Programme / Assessment',
                    'Safe Pass',
                    'Basic Road Strengthening',
                    'Chainsaw Training For Local Authority Operatives - Refresher',
                    'Masonry Arch Repair',
                'CSCS - 360 Degree Excavator Experienced Operator Programme / Assessment',
                    'CSCS - Mini Digger Experienced Operator Programme / Assessment',
                    'Driving Licence Category C',
                    'Driving Licence Category W',
                    'Driving Licence Category C1',
                    'Driving Licence Category CE',
                    'Driving Licence Category C1E',
                    'Driving Licence Category BE',
                    'Safe use of Pesticides & Herbicides - CG',
                    'Safe use of Pesticides & Herbicides - Handheld - CG',
                    'Safe use of Pesticides & Herbicides - Professional users',
                    'Road Services Supervisor',
                    'Road Opening and Reinstatement - Basic',
                    'Road Opening and Reinstatement - Advanced',
                    'First Aid Response',
                    'First Aid Response - Refresher',
                  'Chainsaw Training for New Local Authority Operatives - City & Guilds',
                    'Surface Dressing for Operatives',
                    'Surface Dressing For Engineers',
                    'Surface Dressing - Series 900 Design & Contracts',
                    'Winter Service Operator - Refresher',
                    'CSCS - Site Dumper Experienced Operator Programme / Assessment',
                    'Location of Underground Services - Refresher',
                    'Mini Digger - Refresher',
                    'Site Dumper - Refresher',
                    'Slinger Signaller - Refresher',
                    'Telescopic Handler - Refresher',
'Operation and Maintenance of Lawnmowers, Strimmers, Hedge Trimmers and Leaf Blowers',
                    'School Warden Training',
                    'Safe Loading and Cargo Securing',
                    'Temporary Traffic Management - Inspection and Audit',
                    'Temporary Traffic Management - Level 3 Roads Supervisor',
                    'Temporary Traffic Management on Level 3 Roads: IPV Operative',
                    'Temporary Traffic Management on Level 3 Roads: Mobile Operative',
                    'Temporary Traffic Management on Level 3 Roads: Static Operative',
                  'Temporary Traffic Management Planning & Design - Level 1 and 2 Roads',
                    'Temporary Traffic Management Planning & Design - Level 3 Roads',
                    'Winter Service Management',
                   'Pavement Condition Index Visual Surveyor - Rural and Urban Flexible',

                ]
            },
```

```
environment_section:{
      type:String,
      enum:[
        'Small Stream Risk Scoring',
        'Hazardous Chemical and Spillage Control',
        'Waste Management',
        'Environmental Legislation',
        'Environmental Inspection Skills ',
        'Management of Hazardous Waste in Waste Facilities',
        'Derelict Sites and Dangerous Structures',
        'Site Suitability Assessment for On-Site Waste Water Treatment Systems',
        'Litter Warden Training',
        'Agricultural Pollution Investigation & Inspection',
        'Solid Fuel Regulations',
        'Enforcement of Waste Management Packaging Regulations',
        'Fly Tipped Waste - Safety for Operatives',
        'Courtroom Skills',
        'Management of Invasive Plants and Biosecurity',
        'Aggressive Behaviour - Environmental',
        'Litter Warden Training',
        'Agricultural Pollution Investigation & Inspection',
        'Solid Fuel Regulations',
        'Enforcement of Waste Management Packaging Regulations',
        'Fly Tipped Waste - Safety for Operatives',
        'Courtroom Skills',
        'Management of Invasive Plants and Biosecurity',
        'Aggressive Behaviour - Environmental',
        'Open Source Internet Investigations',
        'Waste Enforcement - Investigation and Prosecution'
      ]
},
equipment_certification:{
    type:String,
    enum:[
    'Lifting Certificate (GA1)',
    'Vehicle Checks',
    'CVRT',
    'Office Equipment PC's/Printers etc…',
    'Power Tools'
]
},
construction_section:{
    type:String,
    enum:[
        'Site Safety Induction',
        'Safety Management for Supervisors',
        'Safe Pass Training',
```

```
            'Risk Assessment & SPA',
            'Chemical Handling Training',
            'Manual Handling',
            'Occupational First Aid',
            'MEWP Training',
            'Boom Hoist Training',
            'Water Safety Course',
            'Fire Extinguisher',
            'Abrasive Wheels',
            'Confined Space Entry / Breathing Apparatus',
            'Confined Space Awareness',
            'Work At Height / Safety Harness',
            'Forklift Driver Training',
            'Side Loader Driver Training',
            'Motorised Pallet Truck Training',
            'Safety Rep.',
            'CSCS 180° Excavator',
            'CSCS Telescopic Handler',
            'CSCS Tractor/Dozer',
            'CSCS Mobile Crane',
            'CSCS 360° Excavator',
            'CSCS Slinger/Signaller',
            'CSCS Articulated Dumper',
            'CSCS Crawler Crane',
            'CSCS Mini Excavator',
            'CSCS Self Erect Tower Crane',
            'CSCS Site Dumper',
            'CSCS Tower Crane',
            'CSCS Roof and Wall Sheeting/Cladding',
            'CSCS Built-Up Roof Felting - Bituminous',
            'CSCS Built-Up Roof Felting  - Single Ply Roofing Systems',
            'CSCS Scaffolding Basic',
            'CSCS Scaffolding Advanced',
            'CSCS Mobile Tower Scaffold',
            'CSCS Locating Underground Services',
            'CSCS Signing, Lighting & Guarding at Roadwork',
            'CSCS Health and Safety Roadworks',
            'CSCS Shot Firing'

        ]
    },

    quarying_section:{
        type:String,
        enum:[
            'Workplace Induction',
            'Safe Pass',
            'Quarry Pass',
```

```
            'Manual Handling',
            'Abrasive Wheels',
            'Occupational First Aid',
            'QSCS 180° Excavator',
            'QSCS Telescopic Handler',
            'QSCS Tractor/Dozer',
            'QSCS Front End Loader',
            'QSCS 360° Excavator',
            'QSCS Slinger/Signaller',
            'QSCS Articulated Dumper',
            'QSCS Crawler Crane',
            'QSCS Mini Excavator',
            'QSCS Rigid Dump Truck',
            'QSCS Site Dumper',
            'QSCS Tower Crane',
            'QSCS Shotfiring ',
            'QSCS Explosives Supervision'
        ]
    },

    administration_section:{
        type:String,
        enum:['Administration']
    },

    agricultural_section:{
        type:String,
        enum:['Agricultural Services & Products']
    },

    automotive_section:{
        type:String,
        enum:['Automotive']
    },

    fishing_section:{
        type:String,
        enum:['Fishing']
    },

    oil_and_gas_section:{
        type:String,
        enum:['Oil & Gas']
    },

    pharmaceutical_section:{
        type:String,
        enum:['Pharmaceutical']
```

```
    },

    retail_and_hospitality:{
        type:String,
        enum:[
            'Manual Handling',
            'Handling Payments',
            'Food Safety In Catering/Food Safety in Catering',
            'Food And Beverage Service',
            'Beverage Product Knowledge',
            'HASAP',
            'Customer Service In The Hospitality And Catering Industry',
            'Safety At Work',
            'Hot Beverage Product Knowledge',
            'Menu Knowledge And Design'
        ]
    },

    haulage_transportation:{
        type:String,
        enum:[
            'Safe Pass',
            'Manual Handling',
            'Forklift Driver Training',
            'Side Loader Driver Training',
            'Motorised Pallet Truck Training',
            'Load Securing',
            'ADR',
            'Driver CPC Module 1 (CVEDT)',
            'Driver CPC Module 2 (MRMET)',
            'Driver CPC Module 3 (HSOPD)',
            'Driver CPC Module 4 (RPDTI)',
            'Driver CPC Module 5 (PROTD)',
            'Driver CPC Module 6 (PROBD)',
            'Driving Licence Category C',
            'Driving Licence Category C1',
            'Driving Licence Category CE',
            'Driving Licence Category C1E',
            'Driving Licence Category BE'
        ]
    },

    security_section:{
        type:String,
        enum:[
            'Safe Pass',
            'Manual Handling',
            'Occupational First Aid',
```

```
            'Guarding Skills QQI Level 4',
            'Door Security Procedures QQI Level 4'
        ]
    },

    healthcare_section:{
        type:String,
        enum:[
            'FETAC Level 5 Certificate in Health Care Support',
            'Manual Handling',
            'Child protection',
            'Violence and aggression',
            'Hand hygiene',
            'Clinical waste awareness/disposal training',
            'Chemical Awareness/safety',
            'Forklift',
            'Waste compactor training',
            'Isolation precaution training',
            'Fire Training',
            'Patient Lifting & Moving',
            'Occupational First Aid'
        ]
    },


    childcare_section:{
        type:String,
        enum:['Childcare']
    },

    forestry_operations:{
        type:String,
        enum:[
            'Manual Handling',
            'Basic First Aid(Lantra)',
            'Safe Pass',
            'CSCS 360° Excavator',
            'Hiab Operator',
            'Driver CPC Module 1 (CVEDT)',
            'Driver CPC Module 2 (MRMET)',
            'Driver CPC Module 3 (HSOPD)',
            'Driver CPC Module 4 (RPDTI)',
            'Driver CPC Module 5 (PROTD)',
            'Driver CPC Module 6 (PROBD)',
            'Manual Handling',
            'Basic First Aid(Lantra)',
            'Safe Pass',
            'CSCS 360° Excavator',
```

```
                'Hiab Operator',
                'Driver CPC Module 1 (CVEDT)',
                'Driver CPC Module 2 (MRMET)',
                'Driver CPC Module 3 (HSOPD)',
                'Driver CPC Module 4 (RPDTI)',
                'Driver CPC Module 5 (PROTD)',
                'Driver CPC Module 6 (PROBD)'
            ]
        },

        expiry_date:{
            type: String
        },
        issue_date:{
            type: String
        },
        file_cert:{
            type: String
        },
        updated:{
            type: Date,
            default: Date.now
        },


        status:{
            type: String,
            enum: ['approved','pending','unapproved'],
            default: 'pending'
        },
        comment:{
            type: String,
            default: Date.now
        }

    });
    module.exports = certSchema;
```

View all certificates

Description:

A User can view the list of asset certificates, includes name, company_name, company_logo in result

- User must be logged in,
- User must be an **admin, employer, institute**
- User must be a follower of **employer**

```
})
   .where('role').equals('employer')
```

```
    .where('followed.follower').equals(user.id)
  ;
```

URL

/certificates/viewall

Method

GET

URL Params

Required:

id = [string], role = [‘admin’, ’employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
[
    {
  "_id": "5e9ee81840847127140ce90f",
  "name": "austin",
  "employerDetails": [
    {
      "company_name": "Construction Concrete",
      "company_logo": "QrTag.png"
    }
  ],
  "assetCerts": [
      {
      "status": "pending",
      "comment": “ ”,
      "updated": "2020-04-25T17:09:57.398Z",
      "_id": "5ea46ee59a8310582432dcb6",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    }
     ]
}
```

```
 ]
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

View Individual certificate by Id (isFollowing() employer)

Description
View asset certificate by Id

- User must be logged in,
- User has to be following the employer,
- User must be **admin, employer, institute**

URL

/certificates/:id/:certID

Method:

GET

URL Params

Required:
id = [string], certID = [string], role = [**admin, employer, institute**]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "pending",
  "comment": “ comment ”,
  "updated": "2020-04-25T16:10:30.315Z",
  "_id": "5ea460f663d53c05d835bf18",
  "assetID": "5ea36437916c505e70e673bd",
  "water_section": "Quality Assurance Wastewater Treatment Plants",
  "expiry_date": "1601679600000",
  "issue_date": "1570057200000",
  "file_cert": "white-image.pdf"
}
```

Error Response:

- **Code:** 401 Unauthorized

Content:

```
{
  "error": "You are not authorized to view this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

View Own individual asset certificate by Id

Description
View Own insurance docs by Id

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

URL

/certificates/myCert/:id:/certID

Method:

GET

URL Params

Required:
id = [string], certID = [string], role = ['employer']

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "pending",
  "comment": " ",
  "updated": "2020-04-25T16:10:30.315Z",
  "_id": "5ea460f663d53c05d835bf18",
  "assetID": "5ea36437916c505e70e673bd",
  "water_section": "Quality Assurance Wastewater Treatment Plants",
  "expiry_date": "1601679600000",
```

```
  "issue_date": "1570057200000",
  "file_cert": "white-image.pdf"
}
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to write this content" }
```


View asset certificates by Asset Id (isFollowing)

Description
View asset certificates related to individual asset

- User must be logged in,
- User has to be following the employer,
- User must be **admin, employer, institute**

URL

/certificates/viewCerts/:id/:certID

Method:

GET

URL Params

Required:
id = [string], certID = [string], role = [**admin, employer, institute**]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  "_id": "5e9ee81840847127140ce90f",
  "assetCerts": [
```

```
    {
      "status": "pending",
      "comment": " ",
      "updated": "2020-04-25T16:10:30.315Z",
      "_id": "5ea460f663d53c05d835bf18",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    },
    {
      "status": "pending",
      "comment": " ",
      "updated": "2020-04-25T16:48:46.135Z",
      "_id": "5ea469ee11332e4b8cb9bc38",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    }
  ] }
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to view this content" }
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


View asset certificates by Asset Id

Description

View asset certificates related to individual asset

- User has to have same id,
- The assets id must be included as a param
- User must be **employer**

URL

/certificates/viewMyCerts/:id/:certID

Method:

GET

URL Params

Required:

id = [string], certID = [string], role = [**employer**]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  "_id": "5e9ee81840847127140ce90f",
  "assetCerts": [
    {
      "status": "pending",
      "comment": " ",
      "updated": "2020-04-25T16:10:30.315Z",
      "_id": "5ea460f663d53c05d835bf18",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    },
    {
      "status": "pending",
      "comment": " ",
      "updated": "2020-04-25T16:48:46.135Z",
      "_id": "5ea469ee11332e4b8cb9bc38",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    }
  ] }
```


Error Response:

- **Code:** 401 Unauthorized

Content:

```
{
  "error": "You are not authorized to view this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Add New Asset certificate

Description:

A logged in User can add a new asset certificate

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

```
employer.assetCerts.push({
             assetID:  req.params.assetID,

             // each of the following fields contains a required
             // enum value, refer to /models/certificate file
            water_section: req.body.water_section,
            roads_section: req.body.roads_section,
            environment_section: req.body.environment_section,
            equipment_certification: req.body.equipment_certification,
            construction_section: req.body.construction_section,
            quarying_section: req.body.quarying_section,
            administration_section: req.body.administration_section,
            agricultural_section: req.body.agricultural_section,
            automotive_section: req.body.automotive_section,
            fishing_section: req.body.fishing_section,
            oil_and_gas_section: req.body.oil_and_gas_section,
            pharmaceutical_section: req.body.pharmaceutical_section,
            retail_and_hospitality: req.body.retail_and_hospitality,
            haulage_transportation: req.body.haulage_transportation,
            security_section: req.body.security_section,
            healthcare_section: req.body.healthcare_section,
            childcare_section: req.body.childcare_section,
            forestry_operations: req.body.forestry_operations,
```

```
            expiry_date:  Date.parse(req.body.expiry_date),
            issue_date:  Date.parse(req.body.issue_date),
            file_cert: file_cert

                });
```

issue_date and expiry_date use Date.parse() to convert the date string to a number value. The number value can be manipulated later in the front end.

issue_date and expiry_date should use an input value of type =”datetime-local”

```
<input name="" type="datetime-local" >
```

Date.parse info: https://www.w3schools.com/jsref/jsref_parse.asp

datetime-local info: https://www.w3schools.com/tags/att_input_type_datetime-local.asp

Method:
PUT

URL

/certificates/add/:id/:assetID

URL Params

Required:
id = [string], assetID = [string], role = [‘employer’]

Data Params

Json

Success Response:

- **Code:** 200
  Content:


```
  {
"_id": "5e9ee81840847127140ce90f",
  "assetCerts": [
    {
      "status": "pending",
      "comment": " ",
      "updated": "2020-04-25T16:10:30.315Z",
      "_id": "5ea460f663d53c05d835bf18",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
```

```
    }
  ]   "__v": 1 }
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."
}
```

Edit Asset certificate

Description:

A logged in User can edit existing asset certificate

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

This route allows for one or all of the following fields to be updated:

- water_section
- roads_section
- environment_section
- equipment_certification
- construction_section
- quarying_section

- administration_section
- agricultural_section
- automotive_section
- fishing_section
- oil_and_gas_section
- pharmaceutical_section
- retail_and_hospitality
- haulage_transportation
- security_section
- healthcare_section
- childcare_section
- forestry_operations
- file_name
- issue_date
- expiry_date

Method:

PUT

URL

/certificates/edit/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employer’]

Data Params

Json

Success Response:

- **Code:** 200
  Content:

```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Insurance brand Updated",
  "file_type": "QrTag.png",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"
}
```


Error Response

- **Code:** 401 Unauthorized

  Content:


```
{
        "error": "You are not authorized to view this content"
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No user found."
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No Employer Details found."
}
```


Update Asset certificate file_type (samplefile express file upload)

Description:

A logged in User can update existing certificates file_type. Use sampleFile instead of company file_type on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
```

```
</div>
```

- User must be logged in,
- User has to have the same Id,
- User must be an **employer**

Method:

PUT

URL

/certificates/updateFile/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employer’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
"_id": "5e9ee81840847127140ce90f",
  "assetCerts": [
    {
      "status": "pending",
      "comment": " ",
      "updated": "2020-04-25T16:10:30.315Z",
      "_id": "5ea460f663d53c05d835bf18",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
      "file_cert": "white-image.png"
    }
  ]   "__v": 1     }
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."
}
```

Delete Asset certificate

Description

Delete Asset certificate by Id,

- User must be logged in,
- User has to have the same Id,
- User must be **employer**

URL

/certificates/delete/:id/:docID

Method

PUT

URL Params

Required:
id = [string], docID = [string], role = [‘employer’]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
      "n": 1,
      "nModified": 0,
      "ok": 1
}
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
{
  "error": "You are not authorized to view this content"
}
```


- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


Review Asset certificate

Description:

The purpose of this route is to allow a user that is a follower of the route :id write access to certain fields. The user can verify the document by changing the **status** field to one of the following enum values [**'approved', 'pending', 'unapproved'**] or the user may alter the **comment** field. For a User to edit an existing certificate

- User must be logged in,
- User has to be a follower of the Id,
- User must be an **admin, employer, institute**

Method:

PUT

URL

/certificates/review/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘admin’, ‘employer’, ‘institute’]

Data Params

Json

Success Response:

- **Code:** 200
  Content:

```
{
  "status": "approved",
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Insurance brand Updated",
  "file_type": "QrTag.png",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"
}
```

Error Response

- **Code:** 401 Unauthorized
  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No Employer Details found."       }
```

- **Code:** 500 Internal Server Error
  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

employee.js

models\employee.js

```
// certificates Schema
let employeeSchema = mongoose.Schema({

    firstname:{
        type: String,
        required: true
    },
    lastname:{
        type: String,
        required: true
    },
    mobile_phone:{
        type: String,
        required: true
    },
    photo_id:{
        type: String,
        required: true
    },
    status:{
        type:String,
        enum:['Active','Inactive'],
```

```
        default: 'Inactive'
    },
    QR_Tag:{
        type: String
    }

});

module.exports = employeeSchema;
```

View all employees

Description:

A User can view the list of employees

- User must be logged in,
- User must be **admin, employer, institute**

URL

/employees/viewall

Method

GET

URL Params

Required:

id = [string], role = [‘admin’, ‘employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
[
{
  "_id": "5e9ee7fe40847127140ce90e",
  "name": "jack",
  "email": "jack@email.com",
  "employment": [
    {
      "_id": "5e9f0abe64ed714e1015c7b2",
      "employer": "5e9ee81840847127140ce90f",
```

```
      "status": "pending",
      "updatedAt": "2020-04-21T15:01:18.248Z",
      "createdAt": "2020-04-21T15:01:18.248Z"
    },
    {
      "_id": "5e9f4c3e9da7814a68051386",
      "employer": "5e9efe47263eb40fd0125b3b",
      "status": "pending",
      "updatedAt": "2020-04-21T19:40:46.140Z",
      "createdAt": "2020-04-21T19:40:46.140Z"
    }
  ],
  "employeeDetails": [
    {
      "status": "Inactive",
      "_id": "5e9f1b3e230d3340204283c3",
      "firstname": "Jack",
      "lastname": "Cunningham",
      "mobile_phone": "4567345",
      "photo_id": "QrTag.png"
    },
    {
      "status": "Inactive",
      "_id": "5ea068b492960d354c2afeef",
      "firstname": "John",
      "lastname": "Cunningham",
      "mobile_phone": "4567345",
      "photo_id": "QrTag.png"
    }
  ]
},
{
  "_id": "5e9efe59263eb40fd0125b3c",
  "name": "john",
  "email": "john@email.com",
  "employment": [
    {
      "_id": "5e9f14fc8725af1c08d2a227",
      "my_employer": {
        "_id": "5e9efe47263eb40fd0125b3b",
        "name": "shane",
        "employerDetails": [
          {
            "company_name": "Construction Concrete",
            "company_logo": "QrTag.png"
          }
        ]
      },
```

```
        "status": "connected",
        "createdAt": "2020-04-21T15:45:00.817Z"
      }
    ],
    "employeeDetails": [
      {
        "status": "Inactive",
        "_id": "5e9f469d2d7f2955b89caf6b",
        "firstname": "John",
        "lastname": "Cunningham",
        "mobile_phone": "4567345",
        "photo_id": "QrTag.png"
      }
    ]
  }
]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:


```
Unauthorized
```


View Employee by Id

Description
View Employee by Id,

- User must be logged in,
- User has to be following the employee,
- User must be **admin, employer, institute**

URL

/employees/:id

Method:

GET

URL Params

Required:
id = [string], role = [‘admin’, ‘employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{

  "_id": "5e9ee7fe40847127140ce90e",
  "name": "jack",
  "email": "jack@email.com",
  "employment": [
    {
      "_id": "5e9f0abe64ed714e1015c7b2",
      "employer": {
        "_id": "5e9ee81840847127140ce90f",
        "name": "austin",
        "employerDetails": [
          {
            "company_name": "Construction Concrete",
            "company_logo": "QrTag.png"
          }
        ]
      },
      "status": "pending",
      "updatedAt": "2020-04-21T15:01:18.248Z",
      "createdAt": "2020-04-21T15:01:18.248Z"
    }
  ],
  "employeeDetails": []
}
```


Error Response:

- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```


- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to write this content" }
```

View Own Employee details by Id

Description
View Own Employee details by Id,

- User must be logged in,
- User has to have the same Id,
- User must be an **employee**

URL

/employees/view_own/:id

Method:

GET

URL Params

Required:
id = [string], role = [‘employee’]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{

  {
  "_id": "5e9efe59263eb40fd0125b3c",
  "name": "john",
  "email": "john@email.com",
  "followed": [
    {
      "follower": {
        "_id": "5e9ee81840847127140ce90f",
        "name": "austin",
        "role": "employer"
      }
    },
    {
      "follower": {
        "_id": "5e9f71816a75f728b4998081",
        "name": "HSE",
        "role": "institute"
      }
```

```
    },
    {
      "follower": {
        "_id": "5e9f6ecd6a75f728b4998080",
        "name": "steve",
        "role": "admin"
      }
    },
    {
      "following": {
        "_id": "5e9f71816a75f728b4998081",
        "name": "HSE",
        "role": "institute",
        "e_learning": [
          {
            "_id": "5ea6b3aac075bf6a9026d54a",
            "media_type": "QrTag.png",
            "subject": "subject update",
            "description": "This is where the course description goes"
          },
          {
            "_id": "5ea6bda42339446cb8304330",
            "media_type": "white-image.png",
            "subject": "subject2",
            "description": "This is where the course description goes"
          }
        ]
      }
    }
  ],
  "employment": [
    {
      "_id": "5e9f14fc8725af1c08d2a227",
      "my_employer": {
        "_id": "5e9efe47263eb40fd0125b3b",
        "name": "shane",
        "employerDetails": [
          {
            "company_name": "Construction Concrete",
            "company_logo": "QrTag.png"
          }
        ]
      },
      "status": "connected",
      "createdAt": "2020-04-21T15:45:00.817Z"
    }
  ],
  "employeeDetails": [
```

```
    {
      "status": "Inactive",
      "_id": "5e9f469d2d7f2955b89caf6b",
      "firstname": "John",
      "lastname": "Cunningham",
      "mobile_phone": "4567345",
      "photo_id": "QrTag.png"
    }
  ]
}}
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to write this content" }
```

Delete employee details

Description

Delete Employee details by Id,

- User must be logged in,
- User has to have the same Id,
- User must be **employee**

URL

/employees/delete/:id/:docID

Method

PUT

URL Params
Required:
id = [string], role = [‘employee’]
Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
      "n": 1,
      "nModified": 0,
      "ok": 1
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content"
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Add New Employee details

Description:

A logged in User can add new employee details { firstname, lastname, sampleFile(photo_id), mobile_phone } 'sampleFile' is used for express fileupload for images and files.

- User must be logged in,
- User has to have the same Id,
- User must be **employee**

Method:

PUT

URL

/employees/add/:id

URL Params

Required:

id = [string], role = [‘employee’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5e9efe59263eb40fd0125b3c",
  "employeeDetails": [
    {
      "status": "Inactive",
      "_id": "5e9f469d2d7f2955b89caf6b",
      "firstname": "John",
      "lastname": "Cunningham",
      "mobile_phone": "4567345",
      "photo_id": "QrTag.png"

}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{ "error": "No Employee Details found." }
```

Edit Employee Details

Description:

A logged in User can edit one or all of the existing employee details (not including photo_id)

Method:

PUT

URL

/employees/edit/:id/:docID

URL Params

Required:
id = [string], docID = [string], role = [‘employer’]

Data Params

Json
Success Response:

- **Code:** 200
  Content:

```
{
  "status": "Inactive",
  "_id": "5e9f469d2d7f2955b89caf6b",
  "firstname": "John",
  "lastname": "Cunningham",
  "mobile_phone": "4567345",
  "photo_id": "QrTag.png"
}
```

Error Response

- **Code:** 401 Unauthorized
  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No Employee Details found."
}
```

- **Code:** 500 Internal Server Error

Content:

```
{
"error": "Error reading user: CastError: Cast to ObjectId failed for value
\"5e9efe593eb40fd0125b3c\" at path \"_id\" for model \"User\""
}
```

Update Employee Photo ID

Description:

A logged in User can edit existing employee Photo ID. Use sampleFile instead of photo_id on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

Requirements:

- User must be logged in,
- User has to have the same Id,
- User must be **employee**

Method:

PUT

URL

/employees/updatePhoto/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employee’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "Inactive",
  "_id": "5e9f469d2d7f2955b89caf6b",
```

```
  "firstname": "John",
  "lastname": "Cunningham",
  "mobile_phone": "4567345",
  "photo_id": "QrTag.png"
}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employee Details found."
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
"error": "Error reading user: CastError: Cast to ObjectId failed for value
\"5e9efe593eb40fd0125b3c\" at path \"_id\" for model \"User\""
}
```

QR Tag

Description:

A logged in User can add a QR tag to existing employee details. Use sampleFile instead of QR_Tag on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
```

```
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

Requirements:

- User must be logged in,
- User has to have the same Id,
- User must be **employee**

Method:
PUT
URL

/employees/add_QR_Tag/:id/:docID

URL Params

Required:
id = [string], docID = [string], role = [‘employee’]

Data Params

Json
Success Response:

- **Code:** 200

  Content:

```
{

  "_id": "5e9ee81840847127140ce90f",
  "assets": [
    {
      "status": "Inactive",
      "_id": "5ea35e7501d77208c09aff4f",
      "asset_type": "Automotive",
      "asset_name": "Lorry",
      "asset_image": "white-image.png",
      "QR_Tag": "QrTag.png"
    }
      ]            }
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Asset Details found."
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
"error": "Error reading user: CastError: Cast to ObjectId failed for value
\"5e9efe593eb40fd0125b3c\" at path \"_id\" for model \"User\""
}
```

employeeCerts.js

models\employeeCert.js

```
// Schema for employee certificates
let empCertsSchema = mongoose.Schema({

    water_section:{
        type: String,
        enum:[
          'Risk Assessment for Cryptosporidium',
          'Filters Evaluation Operation Maintenance',
          'Water Treatment - Dealing with Problems',
          'Drinking Water Regulations',
          'Drinking Water Incident Management Training',
          'Sustainable Urban Drainage System Training (SUDS)',
          'EPANET training',
          'Pumps Operation and Maintenance',
          'Water Clarification Process/THM Removal',
          'Drinking Water Regulations EPA Handbook',
          'Fluoridation of Water Supplies',
          'WasteWater Treatment - Plant Operators Assessment Course including Practi-
cal assessment',
```

```
          'Water Treatment - Plant Operators Assessment Course including Practical as-
sessment',
          'Distribution System Operations and Maintenance',
          'Water Treatment & Distribution - Appreciation',
          'Fats Oil and Grease',
          'Quality Assurance Water Treatment Plants',
          'Quality Assurance Wastewater Treatment Plants',
          'Chlorine Handling',
          'DWNMP Sampling Procedures',
          'Membrane Technology in Water / Wastewater Plants',
          'Operation and Maintenance of Small Wastewater Plants',
          'Safety for Water/Wastewater Workers',
          'Sludge Handling',
          'Taste and Odour Issues in Water Treatment',
          'Troubleshooting the Activated Sludge Process',
          'Water Conservation -Network Management - Leakage Control - Operatives',
          'Water Conservation - Network Management Leakage Control - Managers',
          'Pipeline Corrosion/Lead Services',
          'Confined Spaces - High Risk',
          'Confined Spaces - Low Risk',
          'Confined Spaces - Medium Risk',
          'Confined Spaces - Management of Risk',
          'Confined Spaces - Emergency & Rescue',
          'Distribution System - Unidirectional Flushing',
          'Water Metering and Installation',
          'Hygiene in Water Services',
          'Inspection of Domestic Waste Water Treatment Systems',
          'Management and Operation of Waste Water Overflows',
          'Nutrient Removal in Wastewater Treatment Plants',
          'Control Septicity in Rising Mains & Wastewater networks',
          'Electrofusion'
        ]

    },

    roads_section:{ type:String,
            enum:[
                'CSCS - Mini Digger New Entrant Programme',
                'CSCS - Health & Safety at Roadworks',
                'CSCS - Tractor Dozer New Entrant Programme',
                'CSCS - Locating Underground Services',
                'Trench Support',
                'Driver CPC Module 1 (CVEDT)',
                'Driver CPC Module 2 (MRMET)',
                'Driver CPC Module 3 (HSOPD)',
                'Driver CPC Module 4 (RPDTI)',
                'Driver CPC Module 5 (PROTD)',
                'Driver CPC Module 6 (PROBD)',
```

```
                'Manual Handling Outdoor ',
                'CSCS - 360 Degree Excavator New Entrant Programme',
                'IOSH - Managing Safely for Construction Managers ',
                'Ride on Mower',
                'Operation & Maintenance of Lawnmowers',
                'CSCS - Slinger Signaller New Entrant Programme',
                'CSCS - Signing Lighting & Guarding at Roadworks',
                'Operation and Maintenance of Strimmers & Bushcutters',
                'Safe Systems of Work Plans Training',
                'CSCS - 180 Degree Excavator New Entrant Programme',
                'CSCS - Site Dumper New Entrant Programme',
              'CSCS - 180 Degree Excavator Experienced Operator Programme / Assessment',
                'Safe Pass',
                'Basic Road Strengthening',
                'Chainsaw Training For Local Authority Operatives - Refresher',
                'Masonry Arch Repair',
             'CSCS - 360 Degree Excavator Experienced Operator Programme / Assessment',
                'CSCS - Mini Digger Experienced Operator Programme / Assessment',
                'CSCS - Mini Digger New Entrant Programme',
                'CSCS - Health & Safety at Roadworks',
                'CSCS - Tractor Dozer New Entrant Programme',
                'CSCS - Locating Underground Services',
                'Trench Support',
                'Driver CPC Module 1 (CVEDT)',
                'Driver CPC Module 2 (MRMET)',
                'Driver CPC Module 3 (HSOPD)',
                'Driver CPC Module 4 (RPDTI)',
                'Driver CPC Module 5 (PROTD)',
                'Driver CPC Module 6 (PROBD)',
                'Manual Handling Outdoor ',
                'CSCS - 360 Degree Excavator New Entrant Programme',
                'IOSH - Managing Safely for Construction Managers ',
                'Ride on Mower',
                'Operation & Maintenance of Lawnmowers',
                'CSCS - Slinger Signaller New Entrant Programme',
                'CSCS - Signing Lighting & Guarding at Roadworks',
                'Operation and Maintenance of Strimmers & Bushcutters',
                'Safe Systems of Work Plans Training',
                'CSCS - 180 Degree Excavator New Entrant Programme',
                'CSCS - Site Dumper New Entrant Programme',
               'CSCS - 180 Degree Excavator Experienced Operator Programme / Assessment',
                'Safe Pass',
                'Basic Road Strengthening',
                'Chainsaw Training For Local Authority Operatives - Refresher',
                'Masonry Arch Repair',
              'CSCS - 360 Degree Excavator Experienced Operator Programme / Assessment',
                'CSCS - Mini Digger Experienced Operator Programme / Assessment',
                'Driving Licence Category C',
```

```
            'Driving Licence Category W',
            'Driving Licence Category C1',
            'Driving Licence Category CE',
            'Driving Licence Category C1E',
            'Driving Licence Category BE',
            'Safe use of Pesticides & Herbicides - CG',
            'Safe use of Pesticides & Herbicides - Handheld - CG',
            'Safe use of Pesticides & Herbicides - Professional users',
            'Road Services Supervisor',
            'Road Opening and Reinstatement - Basic',
            'Road Opening and Reinstatement - Advanced',
            'First Aid Response',
            'First Aid Response - Refresher',
            'Chainsaw Training for New Local Authority Operatives - City & Guilds',
            'Surface Dressing for Operatives',
            'Surface Dressing For Engineers',
            'Surface Dressing - Series 900 Design & Contracts',
            'Winter Service Operator - Refresher',
            'CSCS - Site Dumper Experienced Operator Programme / Assessment',
            'Location of Underground Services - Refresher',
            'Mini Digger - Refresher',
            'Site Dumper - Refresher',
            'Slinger Signaller - Refresher',
            'Telescopic Handler - Refresher',
'Operation and Maintenance of Lawnmowers, Strimmers, Hedge Trimmers and Leaf Blowers',
            'School Warden Training',
            'Safe Loading and Cargo Securing',
            'Temporary Traffic Management - Inspection and Audit',
            'Temporary Traffic Management - Level 3 Roads Supervisor',
            'Temporary Traffic Management on Level 3 Roads: IPV Operative',
            'Temporary Traffic Management on Level 3 Roads: Mobile Operative',
            'Temporary Traffic Management on Level 3 Roads: Static Operative',
            'Temporary Traffic Management Planning & Design - Level 1 and 2 Roads',
            'Temporary Traffic Management Planning & Design - Level 3 Roads',
            'Winter Service Management',
            'Pavement Condition Index Visual Surveyor - Rural and Urban Flexible',

        ]
 },

 environment_section:{
       type:String,
       enum:[
         'Small Stream Risk Scoring',
         'Hazardous Chemical and Spillage Control',
         'Waste Management',
         'Environmental Legislation',
         'Environmental Inspection Skills ',
```

```
            'Management of Hazardous Waste in Waste Facilities',
            'Derelict Sites and Dangerous Structures',
            'Site Suitability Assessment for On-Site Waste Water Treatment Systems',
            'Litter Warden Training',
            'Agricultural Pollution Investigation & Inspection',
            'Solid Fuel Regulations',
            'Enforcement of Waste Management Packaging Regulations',
            'Fly Tipped Waste - Safety for Operatives',
            'Courtroom Skills',
            'Management of Invasive Plants and Biosecurity',
            'Aggressive Behaviour - Environmental',
            'Litter Warden Training',
            'Agricultural Pollution Investigation & Inspection',
            'Solid Fuel Regulations',
            'Enforcement of Waste Management Packaging Regulations',
            'Fly Tipped Waste - Safety for Operatives',
            'Courtroom Skills',
            'Management of Invasive Plants and Biosecurity',
            'Aggressive Behaviour - Environmental',
            'Open Source Internet Investigations',
            'Waste Enforcement - Investigation and Prosecution'
        ]
    },
    equipment_certification:{
        type:String,
        enum:[
        'Lifting Certificate (GA1)',
        'Vehicle Checks',
        'CVRT',
        'Office Equipment PC’s/Printers etc…',
        'Power Tools'
    ]
    },
    construction_section:{
        type:String,
        enum:[
            'Site Safety Induction',
            'Safety Management for Supervisors',
            'Safe Pass Training',
            'Risk Assessment & SPA',
            'Chemical Handling Training',
            'Manual Handling',
            'Occupational First Aid',
            'MEWP Training',
            'Boom Hoist Training',
            'Water Safety Course',
            'Fire Extinguisher',
            'Abrasive Wheels',
```

```
            'Confined Space Entry / Breathing Apparatus',
            'Confined Space Awareness',
            'Work At Height / Safety Harness',
            'Forklift Driver Training',
            'Side Loader Driver Training',
            'Motorised Pallet Truck Training',
            'Safety Rep.',
            'CSCS 180° Excavator',
            'CSCS Telescopic Handler',
            'CSCS Tractor/Dozer',
            'CSCS Mobile Crane',
            'CSCS 360° Excavator',
            'CSCS Slinger/Signaller',
            'CSCS Articulated Dumper',
            'CSCS Crawler Crane',
            'CSCS Mini Excavator',
            'CSCS Self Erect Tower Crane',
            'CSCS Site Dumper',
            'CSCS Tower Crane',
            'CSCS Roof and Wall Sheeting/Cladding',
            'CSCS Built-Up Roof Felting - Bituminous',
            'CSCS Built-Up Roof Felting  - Single Ply Roofing Systems',
            'CSCS Scaffolding Basic',
            'CSCS Scaffolding Advanced',
            'CSCS Mobile Tower Scaffold',
            'CSCS Locating Underground Services',
            'CSCS Signing, Lighting & Guarding at Roadwork',
            'CSCS Health and Safety Roadworks',
            'CSCS Shot Firing'

        ]
    },

    quarying_section:{
        type:String,
        enum:[
            'Workplace Induction',
            'Safe Pass',
            'Quarry Pass',
            'Manual Handling',
            'Abrasive Wheels',
            'Occupational First Aid',
            'QSCS 180° Excavator',
            'QSCS Telescopic Handler',
            'QSCS Tractor/Dozer',
            'QSCS Front End Loader',
            'QSCS 360° Excavator',
            'QSCS Slinger/Signaller',
```

```
            'QSCS Articulated Dumper',
            'QSCS Crawler Crane',
            'QSCS Mini Excavator',
            'QSCS Rigid Dump Truck',
            'QSCS Site Dumper',
            'QSCS Tower Crane',
            'QSCS Shotfiring ',
            'QSCS Explosives Supervision'
        ]
    },

    administration_section:{
        type:String,
        enum:['Administration']
    },

    agricultural_section:{
        type:String,
        enum:['Agricultural Services & Products']
    },

    automotive_section:{
        type:String,
        enum:['Automotive']
    },

    fishing_section:{
        type:String,
        enum:['Fishing']
    },

    oil_and_gas_section:{
        type:String,
        enum:['Oil & Gas']
    },

    pharmaceutical_section:{
        type:String,
        enum:['Pharmaceutical']
    },

    retail_and_hospitality:{
        type:String,
        enum:[
            'Manual Handling',
            'Handling Payments',
            'Food Safety In Catering/Food Safety in Catering',
            'Food And Beverage Service',
```

```
            'Beverage Product Knowledge',
            'HASAP',
            'Customer Service In The Hospitality And Catering Industry',
            'Safety At Work',
            'Hot Beverage Product Knowledge',
            'Menu Knowledge And Design'
        ]
    },

    haulage_transportation:{
        type:String,
        enum:[
            'Safe Pass',
            'Manual Handling',
            'Forklift Driver Training',
            'Side Loader Driver Training',
            'Motorised Pallet Truck Training',
            'Load Securing',
            'ADR',
            'Driver CPC Module 1 (CVEDT)',
            'Driver CPC Module 2 (MRMET)',
            'Driver CPC Module 3 (HSOPD)',
            'Driver CPC Module 4 (RPDTI)',
            'Driver CPC Module 5 (PROTD)',
            'Driver CPC Module 6 (PROBD)',
            'Driving Licence Category C',
            'Driving Licence Category C1',
            'Driving Licence Category CE',
            'Driving Licence Category C1E',
            'Driving Licence Category BE'
        ]
    },

    security_section:{
        type:String,
        enum:[
            'Safe Pass',
            'Manual Handling',
            'Occupational First Aid',
            'Guarding Skills QQI Level 4',
            'Door Security Procedures QQI Level 4'
        ]
    },

    healthcare_section:{
        type:String,
        enum:[
            'FETAC Level 5 Certificate in Health Care Support',
```

```
            'Manual Handling',
            'Child protection',
            'Violence and aggression',
            'Hand hygiene',
            'Clinical waste awareness/disposal training',
            'Chemical Awareness/safety',
            'Forklift',
            'Waste compactor training',
            'Isolation precaution training',
            'Fire Training',
            'Patient Lifting & Moving',
            'Occupational First Aid'
        ]
    },

    childcare_section:{
        type:String,
        enum:['Childcare']
    },

    forestry_operations:{
        type:String,
        enum:[
            'Manual Handling',
            'Basic First Aid(Lantra)',
            'Safe Pass',
            'CSCS 360° Excavator',
            'Hiab Operator',
            'Driver CPC Module 1 (CVEDT)',
            'Driver CPC Module 2 (MRMET)',
            'Driver CPC Module 3 (HSOPD)',
            'Driver CPC Module 4 (RPDTI)',
            'Driver CPC Module 5 (PROTD)',
            'Driver CPC Module 6 (PROBD)',
            'Manual Handling',
            'Basic First Aid(Lantra)',
            'Safe Pass',
            'CSCS 360° Excavator',
            'Hiab Operator',
            'Driver CPC Module 1 (CVEDT)',
            'Driver CPC Module 2 (MRMET)',
            'Driver CPC Module 3 (HSOPD)',
            'Driver CPC Module 4 (RPDTI)',
            'Driver CPC Module 5 (PROTD)',
            'Driver CPC Module 6 (PROBD)'
        ]
    },
```

```
      expiry_date:{
          type: String
      },
      issue_date:{
          type: String
      },
      file_cert:{
          type: String
      },
      updated:{
          type: Date,
          default: Date.now
      },
      status:{
          type: String,
          enum: ['approved','pending','unapproved'],
          default: 'pending'
      },



      comment:{
          type: String,
          default: Date.now}
});
module.exports = empCertsSchema;
```

View all employee certificates

Description:

A User can view the list of asset certificates, includes name, email in result

- User must be logged in,
- User must be an **admin, employer, institute**
- User must be a follower of **employee**

```
})
   .where('role').equals('employee')
    .where('followed.follower').equals(user.id)
   ;
```

URL

/employeeCerts/viewall

Method

GET

URL Params

Required:

id = [string], role = [‘admin’, ’employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
  [
{
    "_id": "5e9efe59263eb40fd0125b3c",
    "name": "john",
    "email": "john@email.com",
    "employeeCerts": [
      {
        "status": "pending",
        "comment": “ ”,
        "updated": "2020-04-23T09:09:05.428Z",
        "_id": "5ea15b31737d7261bc265bb5",
        "forestry_operations": "Driver CPC Module 5 (PROTD)",
        "expiry_date": "1601679600000",
        "issue_date": "1570057200000",
        "file_cert": "white-image.png"
      },
      {
        "status": "unapproved",
        "comment":
          "Please update PDF, poor quality"
        ,
        "updated": "2020-04-23T12:51:33.823Z",
        "_id": "5ea18f558490d65cec9caec7",
        "forestry_operations": "Manual Handling",
        "expiry_date": "1601679600000",
        "issue_date": "1570057200000",
        "file_cert": "QrTag.png"
      }
    ]
  } ]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

View Individual employee certificate by Id (isFollowing() employee)

Description

View employee certificate by Id

- User must be logged in,
- User has to be following the employee,
- User must be **admin, employer, institute**

URL

/employeeCerts/viewCert/:id/:certID

Method:

GET

URL Params

Required:

id = [string], certID = [string], role = [**admin, employer, institute**]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "pending",
  "comment": “ comment ”,
  "updated": "2020-04-25T16:10:30.315Z",
  "_id": "5ea460f663d53c05d835bf18",
  "forestry_operations": "Driver CPC Module 5 (PROTD)",
  "expiry_date": "1601679600000",
  "issue_date": "1570057200000",
  "file_cert": "white-image.pdf"
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

View Own individual employee certificate by Id

Description
View Own certificate by Id

- User must be logged in,
- User has to have the same Id,
- User must be an **employee**

URL

/employeeCerts/view_own/:id:/certID

Method:

GET

URL Params

Required:
id = [string], certID = [string], role = [‘employee’]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "pending",
  "comment": “ comment ”,
  "updated": "2020-04-25T16:10:30.315Z",
  "_id": "5ea460f663d53c05d835bf18",
  "forestry_operations": "Driver CPC Module 5 (PROTD)",
  "expiry_date": "1601679600000",
  "issue_date": "1570057200000",
  "file_cert": "white-image.pdf"
}
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to write this content" }
```

Add New employee certificate

Description:

A logged in User can add a new employee certificate

- User must be logged in,
- User has to have the same Id,
- User must be **employee**

```
employee.employeeCerts.push({


// Cert types, refer to file models/employeeCert.js for required enum val-
ues related to each field
            water_section: req.body.water_section,
            roads_section: req.body.roads_section,
            environment_section: req.body.environment_section,
            equipment_certification: req.body.equipment_certification,
            construction_section: req.body.construction_section,
            quarying_section: req.body.quarying_section,
            administration_section: req.body.administration_section,
            agricultural_section: req.body.agricultural_section,
            automotive_section: req.body.automotive_section,
            fishing_section: req.body.fishing_section,
            oil_and_gas_section: req.body.oil_and_gas_section,
            pharmaceutical_section: req.body.pharmaceutical_section,
            retail_and_hospitality: req.body.retail_and_hospitality,
            haulage_transportation: req.body.haulage_transportation,
            security_section: req.body.security_section,
            healthcare_section: req.body.healthcare_section,
            childcare_section: req.body.childcare_section,
```

```
            forestry_operations: req.body.forestry_operations,

            expiry_date:  Date.parse(req.body.expiry_date),
            issue_date:  Date.parse(req.body.issue_date),
            file_cert: file_cert

                });
```

issue_date and expiry_date use Date.parse() to convert the date string to a number value. The number value can be manipulated later in the front end.

issue_date and expiry_date should use an input value of type =”datetime-local”

```
<input name="" type="datetime-local" >
```

Date.parse info: https://www.w3schools.com/jsref/jsref_parse.asp

datetime-local info: https://www.w3schools.com/tags/att_input_type_datetime-local.asp

Method:
PUT

URL

/employeeCerts/add/:id

URL Params

Required:
id = [string], role = [‘employee’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:


```
  {
"_id": "5e9ee81840847127140ce90f",
  "assetCerts": [
    {
      "status": "pending",
      "comment": " ",
      "updated": "2020-04-25T16:10:30.315Z",
      "_id": "5ea460f663d53c05d835bf18",
      "assetID": "5ea36437916c505e70e673bd",
      "water_section": "Quality Assurance Wastewater Treatment Plants",
      "expiry_date": "1601679600000",
      "issue_date": "1570057200000",
```

```
      "file_cert": "white-image.png"
    }
  ]   "__v": 1 }
```

Error Response

- **Code:** 401 Unauthorized
  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No Employer Details found."
}
```

Edit employee certificate

Description:
A logged in User can edit existing employee certificate

- User must be logged in,
- User has to have the same Id,
- User must be an **employee**

This route allows for one or all of the following fields to be updated:

- water_section
- roads_section
- environment_section
- equipment_certification
- construction_section
- quarying_section

- administration_section
- agricultural_section
- automotive_section
- fishing_section
- oil_and_gas_section
- pharmaceutical_section
- retail_and_hospitality
- haulage_transportation
- security_section
- healthcare_section
- childcare_section
- forestry_operations
- issue_date
- expiry_date

Method:

PUT

URL

/employeeCerts/edit/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employee’]

Data Params

Json

Success Response:

- **Code:** 200
  Content:

```
{
  "status": "approved",
```

```
  "updated": "2020-04-26T14:17:29.342Z",
  "_id": "5ea597f922c8095f4057b564",
  "file_name": "Insurance brand Updated",
  "file_type": "QrTag.png",
  "issue_date": "1553126400000",
  "expiry_date": "1584748800000",
  "comment": "No comment"
}
```


Error Response

- **Code:** 401 Unauthorized

  Content:


```
{
        "error": "You are not authorized to view this content"
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No user found."
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No Certificate Details found."
}
```


Update employee certificate file_cert (samplefile express file upload)

Description:

A logged in User can update an existing certificates file_cert. Use sampleFile instead of file_cert on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

- User must be logged in,
- User has to have the same Id,
- User must be an **employee**

Method:

PUT

URL

/employeeCerts/updateFile/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employee’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "status": "unapproved",
  "comment": "Please update PDF, poor quality",
  "updated": "2020-04-23T12:51:33.823Z",
  "_id": "5ea18f558490d65cec9caec7",
  "forestry_operations": "Manual Handling",
  "expiry_date": "1601679600000",
  "issue_date": "1570057200000",
  "file_cert": "filedoc.pdf"
}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
```

```
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."
}
```

Delete Employee certificate

Description

Delete Employee certificate by Id,

- User must be logged in,
- User has to have the same Id,
- User must be **employee**

URL

/employeeCerts/delete/:id/:docID

Method

PUT

URL Params

Required:
id = [string], docID = [string], role = [‘employee’]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
      "n": 1,
      "nModified": 1,
      "ok": 1
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content"
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Review Employee certificate

Description:

The purpose of this route is to allow a user that is a follower of the route :id write access to certain fields. The user can verify the document by changing the **status** field to one of the following enum values [**'approved', 'pending', 'unapproved'**] or the user may alter the **comment** field. For a User to edit an existing certificate

- User must be logged in,
- User has to be a follower of the Id,
- User must be an **admin, employer, institute**

Method:

PUT

URL

/employeeCerts/review/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = ['admin', 'employer', 'institute']

Data Params

Json

Success Response:

- **Code:** 200

  Content:


```
{
   "status": "unapproved",
  "comment": "Please update PDF, poor quality",
  "updated": "2020-04-23T12:51:33.823Z",
  "_id": "5ea18f558490d65cec9caec7",
  "forestry_operations": "Manual Handling",
  "expiry_date": "1601679600000",
  "issue_date": "1570057200000",
  "file_cert": "filedoc.pdf
}
```


Error Response

- **Code:** 401 Unauthorized

  Content:


```
{
        "error": "You are not authorized to view this content"
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No user found."
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No Certificate Details found."          }
```


- **Code:** 500 Internal Server Error

  Content:


```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

admin.js

models\administration.js

```
// certificates Schema
let adminSchema = mongoose.Schema({

    first_name:{
        type: String,
        required: true
    },
    last_name:{
        type: String,
        required: true
    },
    profile_image:{
        type: String,
        required: true
    }
});

module.exports = adminSchema;
```

View all admins

Description:

A User can view the list of admins

- User must be logged in,
- User must be an **institute**

URL

/admin/viewall

Method

GET

URL Params

Required:

id = [string], role = ['institute']

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
 [
 {

 {
   "_id": "5e9f6ecd6a75f728b4998080",
   "name": "steve",
   "email": "steve@email.com",
   "employment": [
     {
       "_id": "5ea5ef7fcb83124bc080d1aa",
       "my_employer": {
         "_id": "5e9f71816a75f728b4998081",
         "name": "HSE",
         "instituteDetails": [
           {}
         ]
       },
       "status": "connected",
       "createdAt": "2020-04-26T20:30:55.829Z"
     }
   ],
   "adminDetails": [
     {
       "_id": "5e9f7449819be934085eb51d",
       "first_name": "Steve",
       "last_name": "Maule",
       "profile_image": "white-image.png"
     }
   ]
 }
]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

View Admin Details by Id

Description

View Admin by Id,

- User must be logged in,
- User Id has to be equal to the **admin** field **employment.my_employer**

```
"employment": [{
    "_id": {
        "$oid": "5ea5ef7fcb83124bc080d1aa"
    },
    "my_employer": {
        "$oid": "5e9f71816a75f728b4998081"
    },
    "status": "connected",
    "createdAt": {
        "$date": {
            "$numberLong": "1587933055829"
        }
    }
}],
```

- User must be an **institute**

URL

/admin/details/:id

Method:

GET

URL Params

Required:

id = [string], role = ['institute']

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
 {
  "_id": "5e9f6ecd6a75f728b4998080",
  "name": "steve",
  "email": "steve@email.com",
  "followed": [
    {
      "_id": "5ea025a015a2f40d2839f808",
      "following": {
        "_id": "5e9efe59263eb40fd0125b3c",
```

```
      "name": "john",
      "role": "employee",
      "employment": [
        {
          "my_employer": "5e9efe47263eb40fd0125b3b"
        }
      ],
      "employeeDetails": [
        {
          "status": "Inactive",
          "photo_id": "QrTag.png"
        }
      ],
      "employerDetails": []
    },
    "status": "connected",
    "createdAt": "2020-04-22T11:08:16.033Z"
  },
  {
    "_id": "5ea0270815a2f40d2839f80a",
    "recipient": {
      "_id": "5e9efe47263eb40fd0125b3b",
      "name": "shane",
      "role": "employer",
      "employerDetails": [
        {
          "company_name": "Construction Concrete"
        }
      ]
    },
    "status": "requested",
    "updatedAt": "2020-04-22T11:14:16.843Z",
    "createdAt": "2020-04-22T11:14:16.843Z"
  },
  {
    "_id": "5ea1af825d015451b85aa6b0",
    "following": {
      "_id": "5e9ee81840847127140ce90f",
      "name": "austin",
      "role": "employer",
      "employment": [
        {},
        {}
      ],
      "employeeDetails": [],
      "employerDetails": [
        {
          "company_name": "Construction Concrete",
```

```
            "company_logo": "QrTag.png"
          }
        ]
      },
      "status": "connected",
      "createdAt": "2020-04-23T15:08:50.910Z"
    }
  ],
  "employment": [
    {
      "_id": "5ea5ef7fcb83124bc080d1aa",
      "my_employer": {
        "_id": "5e9f71816a75f728b4998081",
        "name": "HSE",
        "role": "institute",
        "instituteDetails": [
          {
            "company_name": "HSE",
            "company_logo": "white-image.png"
          }
        ]
      },
      "status": "connected",
      "createdAt": "2020-04-26T20:30:55.829Z"
    }
  ],
  "adminDetails": [
    {
      "_id": "5e9f7449819be934085eb51d",
      "first_name": "Steve",
      "last_name": "Maule",
      "profile_image": "white-image.png"
    },
    {
      "_id": "5ea603bc566bad5144d753c7",
      "first_name": "Steve",
      "last_name": "Maule",
      "profile_image": "white-image.png"
    },
    {
      "_id": "5ea60598c4b9076580cc08ef",
      "first_name": "Steve",
      "last_name": "Maule",
      "profile_image": "white-image.png"
    }
  ]
}}
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to write this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

View Own Admin Details by Id

Description
View Own Admin details by Id,

- User must be logged in,
- User has to have the same Id,
- User must be an **admin**

URL

/admin/My_details/:id

Method:

GET

URL Params

Required:
id = [string], role = [‘admin]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{

{
  "_id": "5e9f6ecd6a75f728b4998080",
  "name": "steve",
  "email": "steve@email.com",
  "followed": [
    {
      "_id": "5ea025a015a2f40d2839f808",
      "following": {
        "_id": "5e9efe59263eb40fd0125b3c",
        "name": "john",
        "role": "employee",
        "employment": [
          {
            "my_employer": "5e9efe47263eb40fd0125b3b"
          }
        ],
        "employeeDetails": [
          {
            "status": "Inactive",
            "photo_id": "QrTag.png"
          }
        ],
        "employerDetails": []
      },
      "status": "connected",
      "createdAt": "2020-04-22T11:08:16.033Z"
    },
    {
      "_id": "5ea0270815a2f40d2839f80a",
      "recipient": {
        "_id": "5e9efe47263eb40fd0125b3b",
        "name": "shane",
        "role": "employer",
        "employerDetails": [
          {
            "company_name": "Construction Concrete"
          }
        ]
```

```
    },
    "status": "requested",
    "updatedAt": "2020-04-22T11:14:16.843Z",
    "createdAt": "2020-04-22T11:14:16.843Z"
  },
  {
    "_id": "5ea1af825d015451b85aa6b0",
    "following": {
      "_id": "5e9ee81840847127140ce90f",
      "name": "austin",
      "role": "employer",
      "employment": [
        {},
        {}
      ],
      "employeeDetails": [],
      "employerDetails": [
        {
          "company_name": "Construction Concrete",
          "company_logo": "QrTag.png"
        }
      ]
    },
    "status": "connected",
    "createdAt": "2020-04-23T15:08:50.910Z"
  }
],
"employment": [
  {
    "_id": "5ea5ef7fcb83124bc080d1aa",
    "my_employer": {
      "_id": "5e9f71816a75f728b4998081",
      "name": "HSE",
      "role": "institute",
      "instituteDetails": [
        {
          "company_name": "HSE",
          "company_logo": "white-image.png"
        }
      ]
    },
    "status": "connected",
    "createdAt": "2020-04-26T20:30:55.829Z"
  }
],
"adminDetails": [
  {
    "_id": "5e9f7449819be934085eb51d",
```

```
      "first_name": "Steve",
      "last_name": "Maule",
      "profile_image": "white-image.png"
    }
  ]
}    },
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to write this content" }
```

Delete Admin Details by Id

Description

Delete Admin details by Id,

- User must be logged in,
- User has to have the same Id,
- User must be **admin**

URL

/admin/delete/:id

Method

PUT

URL Params
Required:
id = [string], role = [‘admin]
Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
      "n": 1,
      "nModified": 0,
      "ok": 1
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content"
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Add New Admin Details

Description:

A logged in User can add new employee details { first_name, last_name, sampleFile(profile_image) } 'sampleFile' is used for express fileupload for images and files.

- User must be logged in,
- User has to have the same Id,
- User must be **admin**

Method:
PUT

URL

/admin/add/:id

URL Params

Required:
id = [string], role = ['admin]

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5e9f6ecd6a75f728b4998080",
  "adminDetails": [
    {
      "_id": "5ea603bc566bad5144d753c7",
      "first_name": "Steve",
      "last_name": "Maule",
      "profile_image": "QrTag.png"
    }
  ],
  "__v": 1}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
       "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{ "error": "No Employee Details found." }
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Edit Admin Details by ID

Description:

A logged in User can edit existing admin details not including profile_image
Method:
PUT
URL

/admin/edit/:id/:docID

URL Params

Required:
id = [string], docID = [string], role = [‘admin’]

Data Params

Json
Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5ea603bc566bad5144d753c7",
  "first_name": "Steve",
  "last_name": "Maule",
  "profile_image": "white-image.png"
}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
```

```
      "error": "No Admin Details found."
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
"error": "Error reading user: CastError: Cast to ObjectId failed for value
\"5e9efe593eb40fd0125b3c\" at path \"_id\" for model \"User\""
}
```

Update Profile Image

Description:

A logged in User can edit existing employee Photo ID. Use sampleFile instead of profile_image on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

Requirements:

- User must be logged in,
- User has to have the same Id,
- User must be **admin**

Method:
PUT
URL

/admin/updatePhoto/:id/:docID

URL Params

Required:
id = [string], docID = [string], role = [‘admin’]

Data Params

Json
Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5ea603bc566bad5144d753c7",
  "first_name": "Steve",
  "last_name": "Maule",
  "profile_image": "QrTag.png"
}
```

Error Response

- **Code:** 401 Unauthorized
  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No Admin Details found."
}
```

- **Code:** 500 Internal Server Error
  Content:

```
{
"error": "Error reading user: CastError: Cast to ObjectId failed for value
\"5e9efe593eb40fd0125b3c\" at path \"_id\" for model \"User\""
}
```

institution.js

models\institute.js

```
let instituteSchema = mongoose.Schema({

    company_name:{
```

```
        type: String
    },
    business_type:{
        type: String,
        enum:[
            'Water',
            'Roads',
            'Environment',
            'Construction',
            'Quarying',
            'Administration',
            'Agricultural Services & Products',
            'Automotive',
            'Fishing',
            'Oil & Gas',
            'Pharmaceutical',
            'Retail and Hospitality',
            'Haulage & Transportation',
            'Security',
            'Healthcare',
            'Childcare',
            'Forestry',
            'Employee'
        ],
        required: true
    },
    company_logo:{
        type: String
    },

    business_email:{
        type: String
    },
    phone:{
        type: String
    },
    address:{
        type: String
    }

});
module.exports = instituteSchema
```

View all

Description:

A logged in User can view the list of Institutions.

URL

/institution/viewall

Method

GET

URL Params

Required:

JWT (User must be logged in to view)

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
  [{
    "_id": "5e9f71816a75f728b4998081",
    "name": "HSE",
    "email": "HSE@email.com",
    "instituteDetails": [
      {
        "_id": "5e9f71ad6a75f728b4998082",
        "company_name": "HSE",
        "business_type": "Health care",
        "company_logo": "QrTag.png",
        "business_email": "HSE@email.com update",
        "phone": "123456123",
        "address": "address,address,address"
      }
    ]
  }
]
```


Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

View Institute details by Id

Description
View Institute by Id,

- User must be logged in,
- User must have same Id
- User must be **institute**

URL

/institution/view/:id (eg. ‘ / institution /view/5e68cd5507eb1121d80f9cc6 ‘)

Method:

GET

URL Params
Required:
id = [string], role = [‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  "_id": "5e9f71816a75f728b4998081",
  "name": "HSE",
  "email": "HSE@email.com",
  "followed": [
    {
      "_id": "5e9f7a7e40a5691430e2583c",
      "following": {
        "_id": "5e9ee81840847127140ce90f",
        "name": "austin",
        "role": "employer",
        "employment": [
          {},
          {}
        ],
        "employeeDetails": [],
        "employerDetails": [
          {
            "company_name": "Construction Concrete",
            "company_logo": "QrTag.png"
          }
        ]
      },
```

```
    "status": "connected",
    "createdAt": "2020-04-21T22:58:06.953Z"
  },
  {
    "_id": "5ea011cdb53ddb30a4613ae2",
    "following": {
      "_id": "5e9efe59263eb40fd0125b3c",
      "name": "john",
      "role": "employee",
      "employment": [
        {
          "my_employer": "5e9efe47263eb40fd0125b3b"
        }
      ],
      "employeeDetails": [
        {
          "photo_id": "QrTag.png"
        }
      ],
      "employerDetails": []
    },
    "status": "connected",
    "createdAt": "2020-04-22T09:43:41.091Z"
  },
  {
    "_id": "5ea6c563987a324ee0990182",
    "follower": {
      "_id": "5e9f6ecd6a75f728b4998080",
      "name": "steve",
      "role": "admin"
    },
    "status": "connected",
    "createdAt": "2020-04-27T11:43:31.593Z"
  },
  {
    "_id": "5ea6e637225f9672141334f0",
    "follower": {
      "_id": "5e9efe59263eb40fd0125b3c",
      "name": "john",
      "role": "employee"
    },
    "status": "connected",
    "createdAt": "2020-04-27T14:03:35.446Z"
  },
  {
    "_id": "5ea6e66c225f9672141334f2",
    "follower": {
      "_id": "5e9efe59263eb40fd0125b3c",
```

```
      "name": "john",
      "role": "employee"
    },
    "status": "connected",
    "createdAt": "2020-04-27T14:04:28.772Z"
  }
],
"employment": [
  {
    "_id": "5ea5ef7fcb83124bc080d1ab",
    "my_employee": {
      "_id": "5e9f6ecd6a75f728b4998080",
      "name": "steve",
      "role": "admin",
      "adminDetails": [
        {
          "profile_image": "QrTag.png"
        }
      ]
    },
    "status": "connected",
    "createdAt": "2020-04-26T20:30:55.866Z"
  }
],
"instituteDetails": [
  {
    "_id": "5e9f71ad6a75f728b4998082",
    "company_name": "HSE",
    "business_type": "Health care",
    "company_logo": "white-image.png",
    "business_email": "HSE@email.com",
    "phone": "123456123",
    "address": "address,address,address"
  }
],
"e_learning": [
  {
    "_id": "5ea6b3aac075bf6a9026d54a",
    "media_type": "QrTag.png",
    "subject": "subject update",
    "description": "This is where the course description goes"
  },
  {
    "_id": "5ea6bda42339446cb8304330",
    "media_type": "white-image.png",
    "subject": "subject2",
    "description": "This is where the course description goes"
  }
```

```
  ]

}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
Unauthorized
```

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Delete institute details

Description

Delete institute details by Id,

- User must be logged in,
- User has to have the same Id,
- User must be **institute**

URL

/institution/delete/:id

Method

PUT

URL Params
Required:
id = [string], docID = [string], role = [‘institute’]
Data Params

None

Success Response:

- **Code:** 200
  Content:

```
{
      "n": 1,
      "nModified": 0,
      "ok": 1
}
```

Error Response:

- **Code:** 401 Unauthorized
  Content:

```
{
  "error": "You are not authorized to view this content"
}
```

- **Code:** 500 Internal Server Error
  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Add New Institution details

Description:

A logged in User can add new Institute details {business_type, company_name, sample-File(company_logo), business_email, phone, address } ‘sampleFile’ is used for express fileupload for images and files.

- User must be logged in,
- User has to have the same Id,
- User must be **institute**

Method:
PUT

URL

/institution/add/:id

URL Params

Required:
id = [string], role = [‘institute’]

Data Params

Json
Success Response:

- **Code:** 200

  Content:


```
{
"_id": "5e9f71816a75f728b4998081",
"instituteDetails": [
  {
    "_id": "5e9f71ad6a75f728b4998082",
    "company_name": "HSE",
    "business_type": "Health care",
    "company_logo": "QrTag.png",
    "business_email": "HSE@email.com update",
    "phone": "123456123",
    "address": "address,address,address"
  }
],}
```


Error Response

- **Code:** 401 Unauthorized

  Content:


```
{
        "error": "You are not authorized to view this content"
}
```


- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Institute Details found."
}
```

Edit Institution Details

Description:

A logged in User can edit one or all of the existing institute details { business_type, company_name, business_email, phone, address } , company_logo is excluded from this request.

- User must be logged in,
- User has to have the same Id,
- User must be **institute**

Method:
PUT

URL

/institution/edit/:id/:docID

URL Params

Required:
id = [string], docID = [string], role = ['institute']

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5e9f71ad6a75f728b4998082",
  "company_name": "HSE",
  "business_type": "Health care",
  "company_logo": "white-image.png",
  "business_email": "HSE@email.com update",
  "phone": "123456123",
```

```
  "address": "address,address,address"
}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."
}
```

Update Institute Company Logo

Description:

A logged in User can edit existing institute Company Logo. Use sampleFile instead of company company_logo on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

Method:
PUT
URL

/institution/updateLogo/:id/:docID

URL Params

Required:
id = [string], docID = [string], role = [‘institute’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5e9f71ad6a75f728b4998082",
  "company_name": "HSE",
  "business_type": "Health care",
  "company_logo": "QrTag.png",
  "business_email": "HSE@email.com update",
  "phone": "123456123",
  "address": "address,address,address"
}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Employer Details found."
}
```

e_learning.js

models\e_Learn.js

```
// Article Schema  for E-Learning Data
```

```
let e_LearnSchema = mongoose.Schema({

    media_type:{
        type: String,
        required: true
    },
    subject:{
        type: String,
        required: true
    },
    description:{
        type: String,
        required: true
    }
});
module.exports = e_LearnSchema;
```

View all e-learning entries

Description:

A User can view the list of e-learning entries created by an employer or an institute role. The route also includes the added fields in the result

- employerDetails.company_name
- employerDetails.company_logo
- instituteDetails.company_name
- instituteDetails.company_logo
- role

URL

/e_learning/viewall

Method

GET

URL Params

Required:

None

Data Params
None

Success Response:

- **Code:** 200
  Content:


```
[
{
  "_id": "5e9ee81840847127140ce90f",
  "role": "employer",
  "employerDetails": [
    {
      "company_name": "Construction Concrete",
      "company_logo": "QrTag.png"
    }
  ],
  "e_learning": []
},
{
  "_id": "5e9efe47263eb40fd0125b3b",
  "role": "employer",
  "employerDetails": [
    {
      "company_name": "Construction Concrete",
      "company_logo": "QrTag.png"
    }
  ],
  "e_learning": []
},
{
  "_id": "5e9f71816a75f728b4998081",
  "role": "institute",
  "employerDetails": [],
  "instituteDetails": [
    {
      "company_name": "HSE",
      "company_logo": "white-image.png"
    }
  ],
  "e_learning": [
    {
      "_id": "5ea6b3aac075bf6a9026d54a",
      "media_type": "QrTag.png",
      "subject": "subject update",
```

```
        "description": "This is where the course description goes"
      },
      {
        "_id": "5ea6bda42339446cb8304330",
        "media_type": "white-image.png",
        "subject": "subject2",
        "description": "This is where the course description goes"
      }
    ]
  }
```


Error Response:

- **Code:** 422 Unprocessable Entity

  Content:


```
{
 "error": "No details found."
}
```


View Individual e-learning entry by Id

Description
View individual e-learning entry

URL

/e_learning/:id/:certID

Method:

GET

URL Params

Required:
id = [string], certID = [string]

Data Params

None

Success Response:

- **Code:** 200

  Content:


```
{
  "_id": "5ea6bab023d396652809048c",
  "media_type": "white-image.png",
  "subject": "subject2",
  "description": "This is where the course description goes"
```

```
}
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

View Individual e-learning entry by Id (if employee is following)

Description
View individual e-learning entry

- User must be logged in,
- User must be an **employee, institute**
- User must be a follower of **employee** or **institute**

URL

/e_learning/learner/:id/:certID

Method:

GET

URL Params

Required:
id = [string], certID = [string], role[‘employee’]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5ea6bab023d396652809048c",
  "media_type": "white-image.png",
```

```
  "subject": "subject2",
  "description": "This is where the course description goes"
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content" }
```

Error Response:

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Add New e-learning entry

Description:

A logged in User can add a new e-learning entry

- User must be logged in,
- User has to have the same Id,
- User must be **employer** or **institute**

Method:

PUT

URL

/e_learning/add/:id

URL Params

Required:

id = [string], role = [‘employer’, ‘institute’]

Data Params

Json

Success Response:

- **Code:** 200

Content:


```
{
  "_id": "5e9f71816a75f728b4998081",
  "e_learning": [
    {
      "_id": "5ea6b3aac075bf6a9026d54a",
      "media_type": "doc1.pdf",
      "subject": "subject update",
      "description": "This is where the course description goes"
    },
    {
      "_id": "5ea6bda42339446cb8304330",
      "media_type": "file.pdf",
      "subject": "subject2",
      "description": "This is where the course description goes"
    },
    {
      "_id": "5eaf338d8834223c6ce6cb2e",
      "media_type": "white-image.png",
      "subject": "subject3",
      "description": "This is where the course description goes"
    }  ]   "__v": 1 }
```


Error Response

- **Code:** 401 Unauthorized

  Content:


```
{
       "error": "You are not authorized to view this content"
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No user found."
}
```


- **Code:** 422 Unprocessable Entity

  Content:


```
{
      "error": "No Employer Details found."
}
```

Code: 500 Internal Server Error

Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Code: 500 Internal Server Error

Content:

```
{
        "error": "Error reading post: ValidationError: e_learning.0.subject:
Path `subject` is required., e_learning.0.description: Path `description` is
required."
"
}
```

Edit existing e-learning entry

Description:

A logged in User can add a new e-learning entry

- User must be logged in,
- User has to have the same Id,
- User must be **employer** or **institute**

This route allows for one or all of the following fields to be updated:

- subject
- description

Method:

PUT

URL

/e_learning /update/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employer’, ‘institute’]

Data Params

Json

Success Response:

- **Code:** 200
  Content:

```
{
  "_id": "5ea6b3aac075bf6a9026d54a",
  "media_type": "QrTag.png",
  "subject": "subject update",
  "description": "This is where the course description goes, it is a String"
}
```

Error Response

- **Code:** 401 Unauthorized
  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No Certificate Details found."
}
```

Edit existing e-learning entry (admin isFollowing)

Description:

A logged in User can add a new e-learning entry

- User must be logged in,
- User has to follow the :id
- User must be **admin**

This route allows for one or all of the following fields to be updated:

- subject
- description

Method:

PUT

URL

/e_learning /admin_update/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employer’, ‘institute’]

Data Params

Json

Success Response:

- **Code:** 200
  Content:


```
{
  "_id": "5ea6b3aac075bf6a9026d54a",
  "media_type": "QrTag.png",
  "subject": "subject update",
  "description": "This is where the course description goes, it is a String"
}
```


Error Response

- **Code:** 401 Unauthorized
  Content:


```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Certificate Details found."
}
```

Update e-learning media_type (samplefile express file upload)

Description:

A logged in User can update an existing e-learning entry media_type. Use sampleFile instead of media_type on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

- User must be logged in,
- User has to have the same Id,
- User must be an **employer** or **institute**

Method:

PUT

URL

/e_learning/updatefile/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘employer’, ‘institute’]

Data Params

Json

Success Response:

- **Code:** 200

  Content:

```
{
  "_id": "5ea6b3aac075bf6a9026d54a",
  "media_type": "QrTag.png",
  "subject": "subject update",
  "description": "This is where the course description goes"
}
```

Error Response

- **Code:** 401 Unauthorized

  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity

  Content:

```
{
      "error": "No Details found."
}
```

Update e-learning media_type (admin isFollowing())

Description:

A logged in User can update an existing e-learning entry media_type. Use sampleFile instead of media_type on the client-side form eg.

```
<div class="form-group row">
<label for="sampleFile" class="">Image</label>
  <div class="col-md-6">
```

```
      <input name="sampleFile" type="file" class="btn btn-primary"
      required title="You Must Upload an Image to Continue">
  </div>
</div>
```

- User must be logged in,
- User has to be following the id
- User must be an **admin**

Method:

PUT

URL

/e_learning/admin_updatefile/:id/:docID

URL Params

Required:

id = [string], docID = [string], role = [‘admin’]

Data Params

Json

Success Response:

- **Code:** 200
  Content:

```
{
  "_id": "5ea6b3aac075bf6a9026d54a",
  "media_type": "QrTag.png",
  "subject": "subject update",
  "description": "This is where the course description goes"
}
```

Error Response

- **Code:** 401 Unauthorized
  Content:

```
{
        "error": "You are not authorized to view this content"
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No user found."
}
```

- **Code:** 422 Unprocessable Entity
  Content:

```
{
      "error": "No Certificate Details found."          }
```

- **Code:** 500 Internal Server Error
  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```

Delete E-learning entry by Id

Description

Delete E-learning entry by Id,

- User must be logged in,
- User has to have the same Id,
- User must be **employer** or **institute**

URL

/e_learning/delete/:id/:docID

Method

PUT

URL Params

Required:

id = [string], docID = [string], role = [‘employer’, ‘institute’]

Data Params

None

Success Response:

- **Code:** 200

  Content:

```
{
      "n": 1,
      "nModified": 1,
      "ok": 1
}
```

Error Response:

- **Code:** 401 Unauthorized

  Content:

```
{
  "error": "You are not authorized to view this content"
}
```

- **Code:** 500 Internal Server Error

  Content:

```
{
      "error": "Error reading user: CastError: Cast to ObjectId failed for
value \"5e9efe59263eb40fd0\" at path \"_id\" for model \"User\"
"
}
```