For starting any **MERN project** include the following code lines that is necessary-

```
const express = require('express');
const mongoose = require('mongoose');
const multer = require('multer'); //for file upload
const path = require('path');
const session = require('express-session');
const bcrypt = require('bcrypt'); //for password hide in form of ****
const { title } = require('process');
const { name } = require('ejs');
const { type } = require('os');
const { countReset, log } = require('console');
const app = express();
const port= 1000;

//connect to mongodb
mongoose.connect('mongodb://localhost:27017/ERP',{
    useNewUrlParser: true,
    useUnifiedTopology: true
});
```

For login u must need

- Download express, express session.
- Now for upload and also for running express-

```
//  Middleware to parse JSON and URL-encoded data
app.use(express.json());
app.use(express.urlencoded({ extended: true}));
app.use('/uploads', express.static('uploads'));

const storage = multer.diskStorage({
        destination: './uploads/',
        filename: function(req, file, cb){
         cb(null, file.fieldname + '-' + Date.now() +
path.extname(file.originalname));
    }
});
```

- Then serve the form all the forms-

```
//server the form
app.get('/form', (req, res) => {
    res.render('form');
});
```

- Now create a express session-

```
//set up the session managaement
app.use(session({
    secret: 'your_secret_key',
    resave: false,
    saveUninitialized: true
}));
```

- Create a schema for the form enteries-

```
//create a user schema
const userSchema = new mongoose.Schema({
    uuid: {type: String, required: true, unique: true},
    pass: {type: String, required: true},
    college: {type: String, required: true},

});
```

- Create a model to activate the schema-

```
//create model
const Faculty = mongoose.model('Faculty', facultySchema);
const User = mongoose.model('User', userSchema);
```

- for creating a new user who can login-

```
//create a new user
app.post('/student', async (req, res) => {
  try {
            const {uuid, pass, college} = req.body;
            const user = new User({uuid, pass, college});
            await user.save();
            res.status('User registered successfully');
    }catch (error) {
        res.status(400).send(error);
    }
});
```

- Now create a login schema which inspects the info and the login-

```
//create a login

app.post('/log', async (req, res) => {
    try{
        const {uuid, pass, college} = req.body;
        const user = await User.findOne({uuid, college});
        if(user && pass === user.pass){
            req.session.userId = user._id;
            res.redirect('/std');
        }else {
            res.render('Login');

        }
    }catch (error){
        res.status(500).send('Something went wrong');
    }
});
```

- Now after login you need to do logout.

```
//session Logout
app.get('/logout', (req, res) => {
    req.session.destroy((err) => {
        if(err){
            return res.status(500).send('Something went wrong');
        }
        res.redirect('/');
    });
});
```

To run the ejs files we have to create view ejs engine, which also initialized the views folder

```
//connect to engine
app.set('view engine', 'ejs');
```

similarily to initialize the public folder-

```
//serve static file(Css, js images, etc.)
app.use(express.static('public'));
```

now to define the routes of the desired ejs page-

```
app.get('/Login', (req, res) => {
    res.render('Login',{ title:'Home', content:'Welcome to my home page'});
});
```

At last to access the console on the web page-

```
//Console login
app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```