

PUSL 3120
Full Stack Development Module
Referral Coursework

Palle Prasad
10749129

Full Stack Development Module Report

Video Link: <https://youtu.be/2AOvP5eBW7w>

User Authentications and Authorization

This part of the project has been implemented with the aim of developing user authentications for the application.

Two different schemas have been created here.

1. Users

2. Role

Role schema will be used to store all the roles with an ID.

Full Stack

MEAN-AUTH

MEAN-AUTH / Get All Roles

GET http://localhost:8800/api/role/getAll

Params Authorization Headers (6) Body Pre-request Script Tests Settings

None form-data x-www-form-urlencoded raw Binary GraphQL JSON

1

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "_id": "65d51d74bcb0445d3e0e4",
4     "role": "Admin - Dev",
5     "createdAt": "2023-10-04T18:43:00Z",
6     "updatedAt": "2023-10-04T18:42:00Z",
7     "__v": 0
8   },
9   {
10    "_id": "65d51d74bcb0445d3e0e5",
11    "role": "Admin - Test",
12    "createdAt": "2023-10-04T18:43:14Z",
13    "updatedAt": "2023-10-04T18:42:00Z",
14    "__v": 0
15  },
16  {
17    "_id": "65d51d74bcb0445d3e0e6",
18    "role": "User - Dev",
19    "createdAt": "2023-10-04T18:43:22Z",
20    "updatedAt": "2023-10-04T18:42:00Z",
21    "__v": 0
22  },
23  {
24    "_id": "65d51d75151aa430807c20084",
25    "role": "Delete",
26    "createdAt": "2023-10-04T18:43:30Z",
27    "updatedAt": "2023-10-04T18:42:00Z",
28    "__v": 0
29  },
30  {
31    "_id": "65d51d7cc74dab1ff7e0",
32    "role": "Read Only",
33    "createdAt": "2023-10-04T18:43:38Z",
34    "updatedAt": "2023-10-04T18:42:00Z",
35    "__v": 0
36  }
]
```

Status: 200 OK Time: 44 ms Size: 942 B Save as Example

Some examples of roles can be described as Admin, and User.

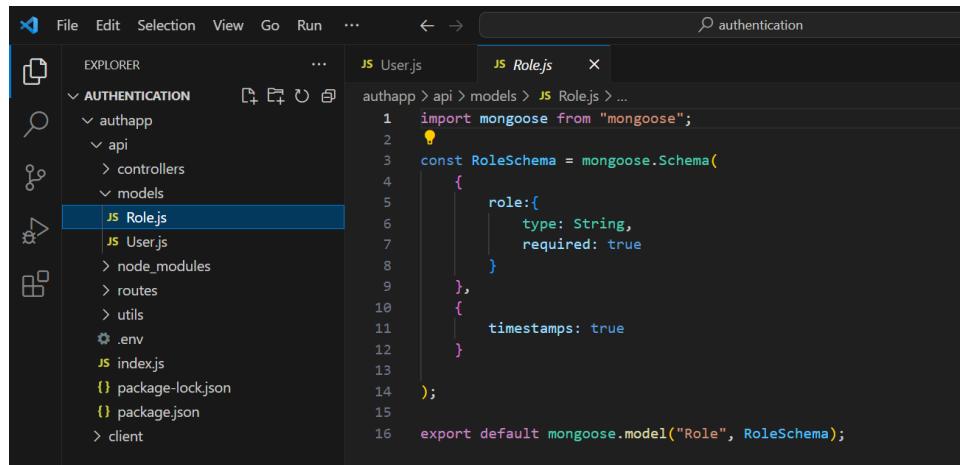
User.js

The screenshot shows a code editor interface with the following details:

- File Explorer (Left):** Shows the project structure:
 - AUTHENTICATION**: authapp, api, controllers, models (Role.js, User.js selected), node_modules, routes, utils, .env
 - index.js
 - package-lock.json
 - package.json
 - client
- Code Editor (Center):** The `User.js` file content is displayed:

```
authapp > api > models > User.js > UserSchema
29   profileImage:{
30     type: String,
31     // required: false,
32     default: ''
33   },
34   isAdmin: {
35     type:Boolean,
36     default: false,
37   },
38   roles:{
39     type: [Schema.Types.ObjectId],
40     required: false,
41     ref: "Role" // refers to the "Role" table in the Role.js
42   }
43 },
44 {
45   timestamps: true // the date and time of creation and updates
46 }
47 };
48 );
49 );
50 };
51
52 export default mongoose.model("User", UserSchema);
53
```
- Status Bar (Bottom):** Ls 5 Col 91 Spaces 2 UTF-8 CR/LF () JavaScript

Role.js file

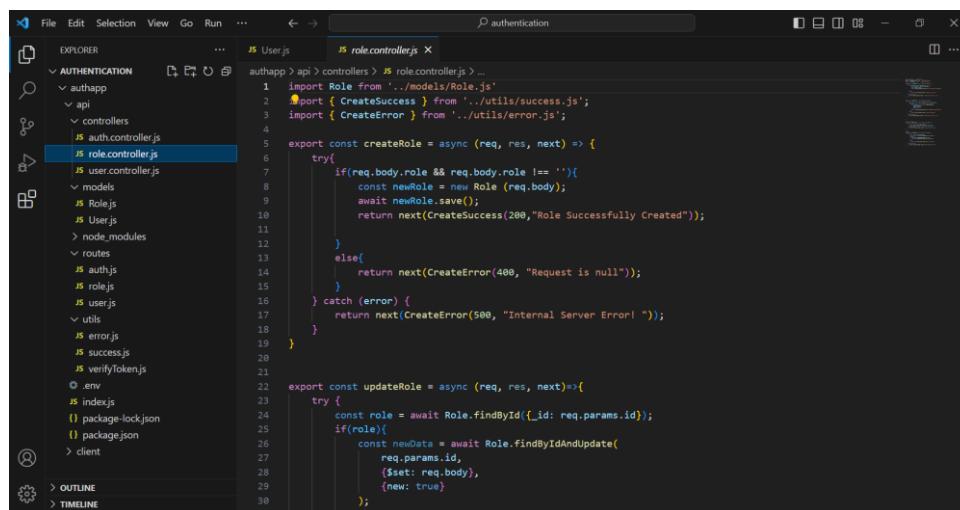


```
authapp > api > models > JS Role.js > ...
1 import mongoose from "mongoose";
2
3 const RoleSchema = mongoose.Schema(
4   {
5     role: {
6       type: String,
7       required: true
8     },
9   },
10  {
11    timestamps: true
12  }
13);
14
15
16 export default mongoose.model("Role", RoleSchema);
```

Creating APIs for Creating, Reading, Updating, Deleting roles

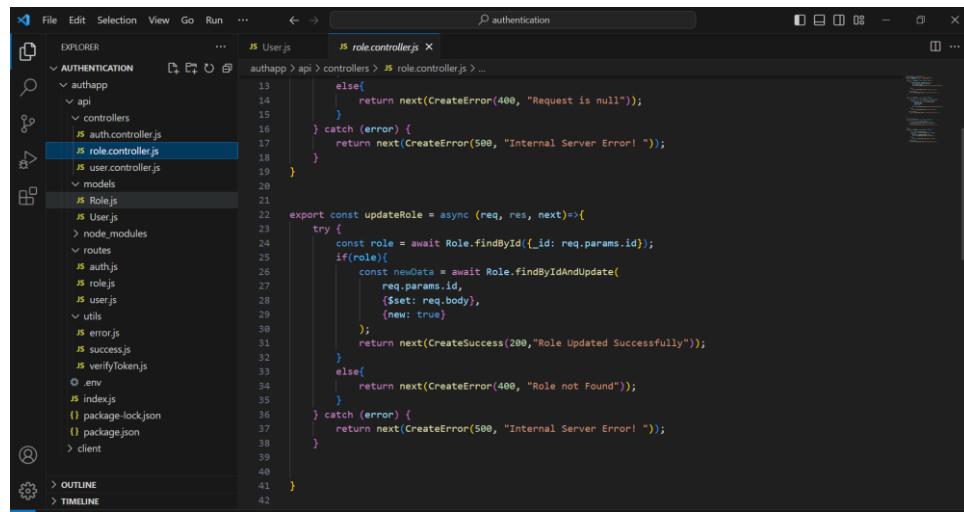
* role.controller.js

Create Role



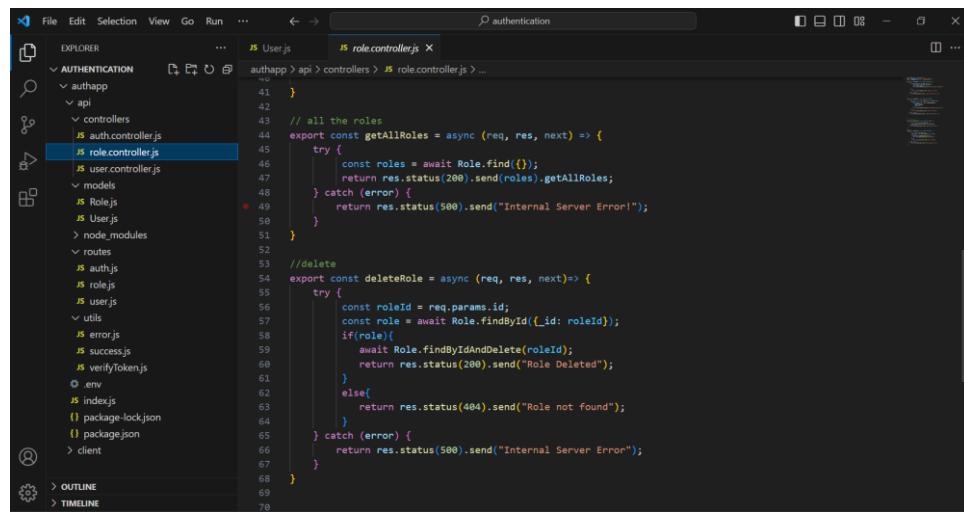
```
authapp > api > controllers > JS role.controller.js > ...
1 import Role from '../models/Role.js';
2 import { CreateSuccess } from './utils/success.js';
3 import { CreateError } from './utils/error.js';
4
5 export const createRole = async (req, res, next) => {
6   try {
7     if(req.body.role && req.body.role != ''){
8       const newRole = new Role (req.body);
9       await newRole.save();
10      return next(CreateSuccess(200,"Role Successfully Created"));
11    }
12    else{
13      return next(CreateError(400, "Request is null"));
14    }
15  } catch (error) {
16    return next(CreateError(500, "Internal Server Error! "));
17  }
18}
19
20
21
22 export const updateRole = async (req, res, next)=>{
23   try {
24     const role = await Role.findById({_id: req.params.id});
25     if(role){
26       const newData = await Role.findByIdAndUpdate(
27         req.params.id,
28         {$set: req.body},
29         {new: true}
30       );
31     }
32   }
33 }
```

Update Role



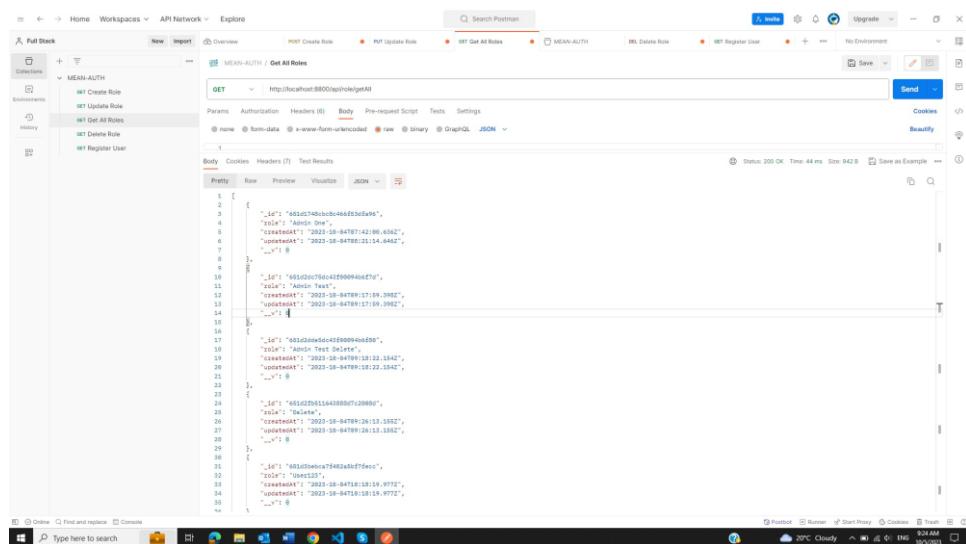
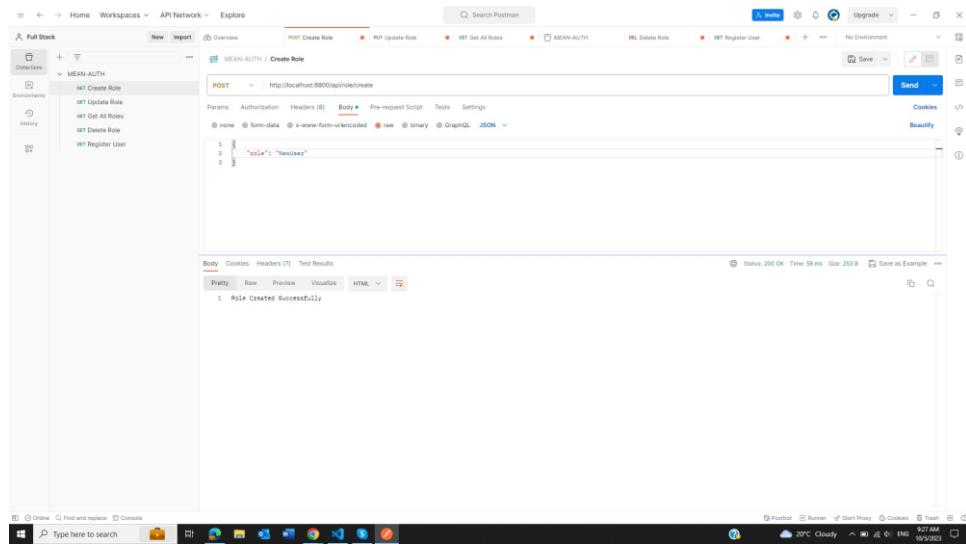
```
13     else{
14         return next(CreateError(400, "Request is null"));
15     }
16 } catch (error) {
17     return next(CreateError(500, "Internal Server Error! "));
18 }
19 }
20
21
22 export const updateRole = async (req, res, next)=>{
23     try {
24         const role = await Role.findById({_id: req.params.id});
25         if(role){
26             const newData = await Role.findByIdAndUpdate(
27                 req.params.id,
28                 {$set: req.body},
29                 {new: true}
30             );
31             return next(CreateSuccess(200,"Role Updated Successfully"));
32         }
33     } catch (error) {
34         return next(CreateError(400, "Role not Found"));
35     }
36 } catch (error) {
37     return next(CreateError(500, "Internal Server Error! "));
38 }
39
40
41 }
```

Get all Roles & Delete Role



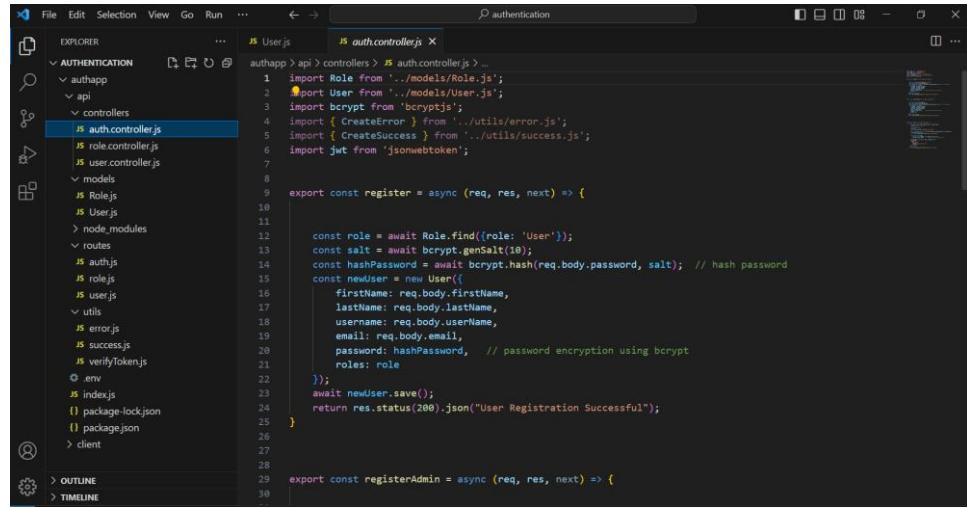
```
41 }
42
43 // all the roles
44 export const getAllRoles = async (req, res, next)=> {
45     try {
46         const roles = await Role.find();
47         res.status(200).send(roles);
48     } catch (error) {
49         return res.status(500).send("Internal Server Error!");
50     }
51 }
52
53 //delete
54 export const deleteRole = async (req, res, next)=> {
55     try {
56         const roleId = req.params.id;
57         const role = await Role.findById({_id: roleId});
58         if(role){
59             await Role.findByIdAndUpdate(roleId);
60             return res.status(200).send("Role Deleted");
61         }
62         else{
63             return res.status(404).send("Role not found");
64         }
65     } catch (error) {
66         return res.status(500).send("Internal Server Error");
67     }
68 }
69
70 }
```

Testing APIs with the use of postman

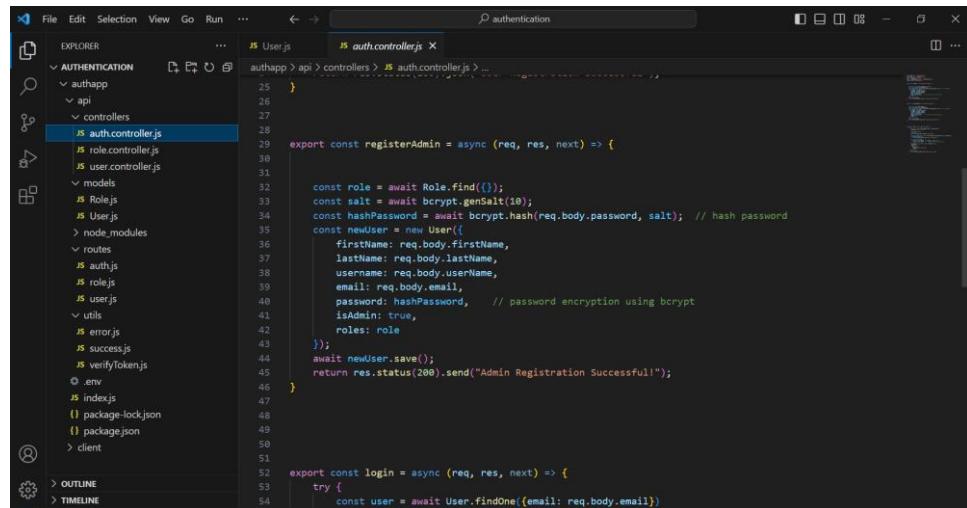


Register, Register as an Admin and Login Functionalities

These functionalities have been implemented inside of the auth.controller.js file which is inside of the controllers folder.



```
File Edit Selection View Go Run ... ← → JS authentication
EXPLORER authapp > api > controllers > auth.controller.js ...
authapp > api > controllers > auth.controller.js > ...
1 import Role from '../models/Role.js';
2 import User from '../models/User.js';
3 import bcrypt from 'bcryptjs';
4 import { CreateError } from '../utils/error.js';
5 import { CreateSuccess } from '../utils/success.js';
6 import jwt from 'jsonwebtoken';
7
8 export const register = async (req, res, next) => {
9
10     const role = await Role.find({role: 'User'});
11     const salt = await bcrypt.genSalt(10);
12     const hashPassword = await bcrypt.hash(req.body.password, salt); // hash password
13     const newUser = new User({
14         firstName: req.body.firstName,
15         lastName: req.body.lastName,
16         username: req.body.userName,
17         email: req.body.email,
18         password: hashPassword, // password encryption using bcrypt
19         roles: role
20     });
21     await newUser.save();
22     return res.status(200).json("User Registration Successful");
23 }
24
25 }
26
27
28
29 export const registerAdmin = async (req, res, next) => {
30
31
32     const role = await Role.find({});
33     const salt = await bcrypt.genSalt(10);
34     const hashPassword = await bcrypt.hash(req.body.password, salt); // hash password
35     const newUser = new User({
36         firstName: req.body.firstName,
37         lastName: req.body.lastName,
38         username: req.body.userName,
39         email: req.body.email,
40         password: hashPassword, // password encryption using bcrypt
41         isAdmin: true,
42         roles: role
43     });
44     await newUser.save();
45     return res.status(200).send("Admin Registration Successful");
46 }
47
48
49
50
51
52 export const login = async (req, res, next) => {
53     try {
54         const user = await User.findOne({email: req.body.email});
```



```
File Edit Selection View Go Run ... ← → JS authentication
EXPLORER authapp > api > controllers > auth.controller.js ...
authapp > api > controllers > auth.controller.js > ...
25 }
26
27
28
29 export const registerAdmin = async (req, res, next) => {
30
31
32     const role = await Role.find({});
33     const salt = await bcrypt.genSalt(10);
34     const hashPassword = await bcrypt.hash(req.body.password, salt); // hash password
35     const newUser = new User({
36         firstName: req.body.firstName,
37         lastName: req.body.lastName,
38         username: req.body.userName,
39         email: req.body.email,
40         password: hashPassword, // password encryption using bcrypt
41         isAdmin: true,
42         roles: role
43     });
44     await newUser.save();
45     return res.status(200).send("Admin Registration Successful");
46 }
47
48
49
50
51
52 export const login = async (req, res, next) => {
53     try {
54         const user = await User.findOne({email: req.body.email});
```

Postman API Test Result:

```

POST http://localhost:8000/api/auth/register-admin
Status: 200 OK Time: 187 ms Size: 258 B
{
  "msg": "Add Registration Successful"
}

```

MongoDB Compass - localhost:27017/AuthDB.users

AuthDB.users

Documents Aggregations Schema Indexes Validation

`_id: 00ec120c011f1bb0fffc0742470aa3e1`

```

{
  "firstname": "John",
  "lastname": "Doe",
  "username": "johndoe",
  "email": "johndoe@doe.com",
  "password": "1234567890",
  "profileImage": null,
  "roles": [
    "admin"
  ],
  "createdAt": "2023-10-05T08:00:00Z",
  "updatedAt": "2023-10-05T08:00:00Z"
}

{
  "id": 00ec120c011f1bb0fffc0742470aa3e1,
  "firstname": "John",
  "lastname": "Doe",
  "username": "johndoe",
  "email": "johndoe@doe.com",
  "password": "1234567890",
  "profileImage": null,
  "roles": [
    "admin"
  ],
  "createdAt": "2023-10-05T08:00:00Z",
  "updatedAt": "2023-10-05T08:00:00Z"
}

```

Testing on Postman

The screenshot shows the Postman application interface. On the left, the 'EXPLORER' panel displays the project structure for 'AUTHAPP' with files like auth.controller.js, role.controller.js, Role.js, User.js, auth.js, role.js, .env, index.js, package-lock.json, and package.json. In the center, the 'MEAN-AUTH / Login user' collection is selected. A POST request is being tested with the URL `http://localhost:8800/api/auth/login`. The 'Body' tab contains the following JSON payload:

```
1: {  
2:   "email": "alizakariah@outlook.com",  
3:   "password": "897654321"  
4: }
```

The 'Test Results' section shows a 400 Bad Request status with a response time of 89 ms. The response body is empty.

This screenshot shows the same Postman session after a successful login. The 'Body' tab now displays the JSON response from the successful login:

```
1: {  
2:   "status": 200,  
3:   "message": "Login Successful",  
4:   "data": {  
5:     "id": "6512097220303e799058284",  
6:     "firstname": "Red",  
7:     "lastname": "Black",  
8:     "username": "rgblack",  
9:     "email": "rgblack@outlook.com",  
10:    "modifiedAt": "2023-10-06T14:58:41Z",  
11:    "createdAt": "2023-10-06T14:58:41Z",  
12:    "role": "User",  
13:    "isDeleted": false,  
14:    "updatedAt": "2023-10-06T04:13:38.907Z",  
15:    "createdAt": "2023-10-06T04:13:38.907Z",  
16:    "id": 8  
17:  }  
18: }
```

The 'Test Results' section shows a 200 OK status with a response time of 103 ms and a size of 840 B. The response body is identical to the JSON above.

User Registration Testing on Postman

The screenshot shows the Postman interface with a successful POST request to the '/auth/register' endpoint. The request body contains a JSON object with fields: first_name, last_name, address, city, state, email, and password. The response status is 200 OK, and the message is "User Registration Successful".

```

{
  "success": true,
  "status": 200,
  "message": "User Registration Successful"
}

```

getAllUser and getById

The screenshot shows the VS Code interface with the 'user.js' file open. The file contains code for a Router that handles '/allUsers' and '/:id'. A terminal window shows the Node.js application starting and connecting to a database.

```

[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Connected to Backend
Database connection is sucessful
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Connected to Backend
Database connection is sucessful

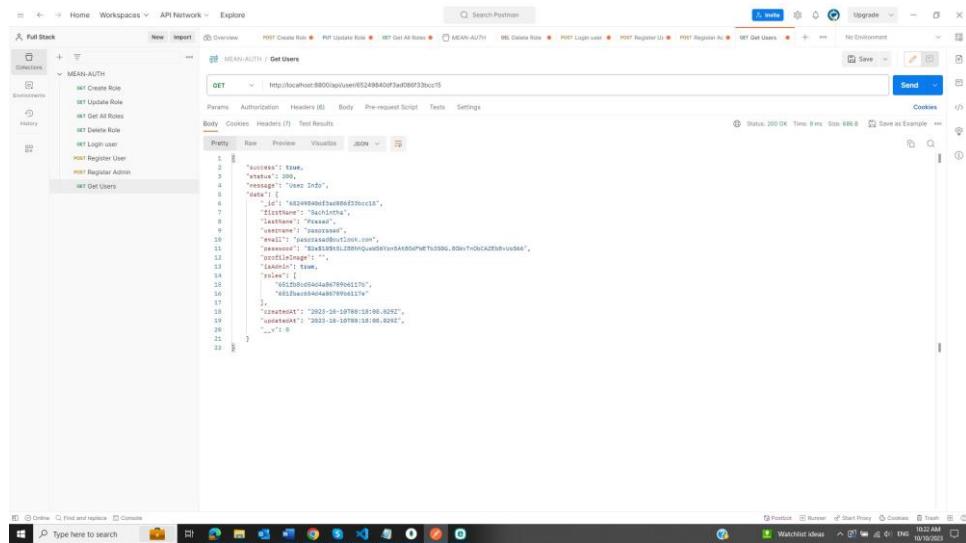
```

The screenshot shows the Postman interface with a successful GET request to the '/allUsers' endpoint. The response status is 200 OK, and it returns a list of users with their IDs, names, and other details.

```

[
  {
    "id": "62fdec1d544e428092199b",
    "firstname": "Aswin",
    "lastname": "A",
    "address": "Kannur",
    "city": "Kannur",
    "state": "Kerala",
    "username": "Aswin123",
    "password": "82a0586c7f5eff7c544998a52a2a098197e5a5f0en7ma100",
    "profileImage": "https://aswin123.com",
    "role": "User",
    "roles": []
  },
  {
    "id": "62fdec1d544e428092199b",
    "firstname": "Aswin",
    "lastname": "A",
    "address": "Kannur",
    "city": "Kannur",
    "state": "Kerala",
    "username": "Aswin123",
    "password": "82a0586c7f5eff7c544998a52a2a098197e5a5f0en7ma100",
    "profileImage": "https://aswin123.com",
    "role": "User",
    "roles": []
  },
  {
    "id": "62fdec1d544e428092199b",
    "firstname": "Aswin",
    "lastname": "A",
    "address": "Kannur",
    "city": "Kannur",
    "state": "Kerala",
    "username": "Aswin123",
    "password": "82a0586c7f5eff7c544998a52a2a098197e5a5f0en7ma100",
    "profileImage": "https://aswin123.com",
    "role": "User",
    "roles": []
  }
]

```



JSON WEB TOKENS

A token will be created only after the login is successful.

JWT Secret key is stored in the .env file

This will create the jwt token for the user:

```

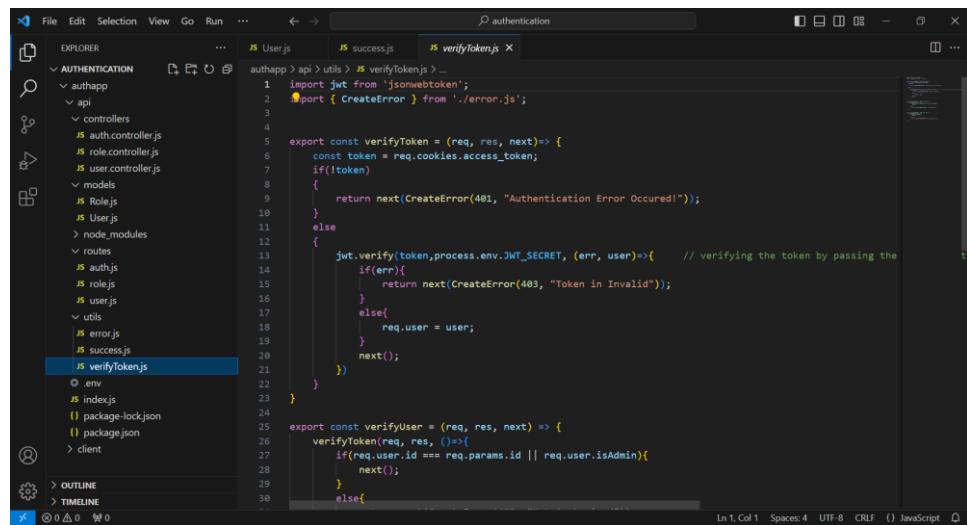
1  export const login = async (req, res, next) => {
2    try {
3      const user = await User.findOne({email: req.body.email})
4      .populate("roles", "role");
5
6      const {roles} = user;
7      if(!user)
8        return res.status(404).send("User Not Found");
9
10     const isPasswordCorrect = await bcrypt.compare(req.body.password, user.password);
11     if(!isPasswordCorrect)
12       return res.status(400).send("Incorrect Password");
13
14     const token = jwt.sign(
15       {id: user._id, isAdmin: user.isAdmin, roles:roles},
16       process.env.JWT_SECRET // secret key in .env config file
17     )
18
19     res.cookie("access_token", token, {httpOnly:true})
20     .status(200)
21     .json({
22       status:200,
23       message: "Login Successful",
24       data:user
25     })
26   } catch (error) {
27     return res.status(500).send("Something went wrong");
28   }
29 }

```

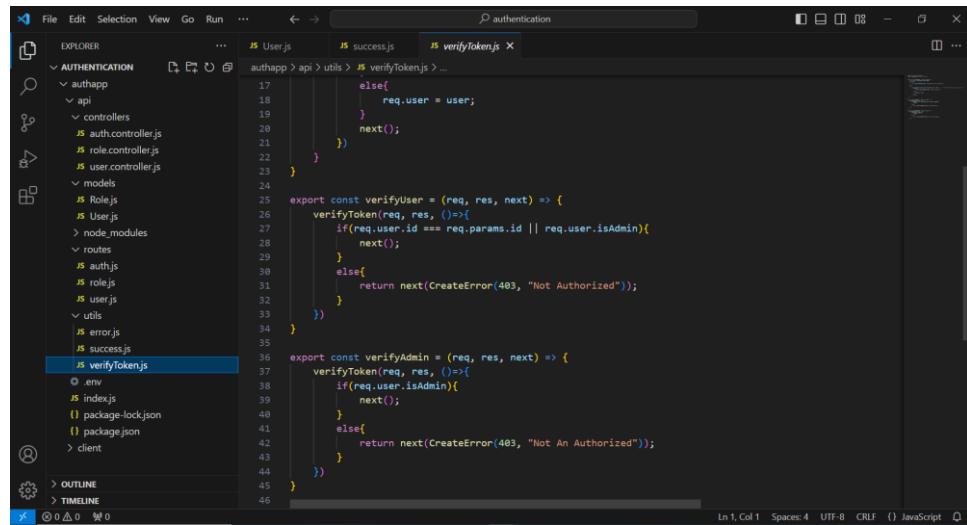
This token will be sent directly to the users using a cookie.

Protecting the API end point using JWT authentication.

Verifying Token(verifyToken.js)



```
File Edit Selection View Go Run ... authentication
EXPLORER ... JS User.js JS success.js JS verifyToken.js ...
authapp > api > utils > JS verifyToken.js ...
1 import jwt from 'jsonwebtoken';
2 import { CreateError } from './error.js';
3
4
5 export const verifyToken = (req, res, next) => {
6   const token = req.cookies.access_token;
7   if(!token)
8   {
9     return next(CreateError(401, "Authentication Error Occurred!"));
10  }
11  else
12  {
13    jwt.verify(token, process.env.JWT_SECRET, (err, user) => { // verifying the token by passing the
14      if(err)
15        return next(CreateError(403, "Token is Invalid"));
16      else{
17        req.user = user;
18      }
19      next();
20    });
21  }
22}
23
24
25 export const verifyUser = (req, res, next) => {
26   verifyToken(req, res, () =>{
27     if(req.user.id === req.params.id || req.user.isAdmin){
28       next();
29     }
30     else{
31       return next(CreateError(403, "Not Authorized"));
32     }
33   });
34 }
35
36
37 export const verifyAdmin = (req, res, next) => {
38   verifyToken(req, res, () =>{
39     if(req.user.isAdmin){
40       next();
41     }
42     else{
43       return next(CreateError(403, "Not An Authorized"));
44     }
45   });
46 }
```



```
File Edit Selection View Go Run ... authentication
EXPLORER ... JS User.js JS success.js JS verifyToken.js ...
authapp > api > utils > JS verifyToken.js ...
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
```

Access token cookie

The screenshot shows the Postman application interface. A collection named 'MEAN-AUTH' is selected, containing a 'Register User' request. The request is a POST to `http://localhost:8000/api/auth/register/`. The JSON body contains user registration data:

```
[{"firstName": "Rowan", "lastName": "Atkinson", "password": "12345678", "email": "rowan@cello.com", "password": "12345678"}]
```

The response status is 200 OK, with a response time of 177 ms and a size of 305 B. A new cookie, `access_token`, is listed in the Cookies tab, with its value being a long string of characters.

Using jwt.io to decode the encoded tokens



Algorithm

Encoded

```
eyJhbGciOiJIUzI1NiIsIn  
R5cCI6IkpXVCJ9.eyJpZCI  
6IjY1MjQ5ODQwZGYzYWQwO  
DZmMzNiY2MxNSIsImlzQWR  
taW4iOnRydWUsInJvbGVzI  
jpbeJfaWQiOiiI2NTFmYjh  
jZDU0ZDRhMDY3MDliNjExN  
2IiLCJyb2xIjoiQWRtaW4  
ifSx7I19pZCI6IjY1MWZiY  
WM2NTRkNGEwNjcwOWI2MTE  
3ZSIIsInJvbGUiOijVc2Vyi  
n1dLCJpYXQiOjE2OTY5MDQ  
4NjJ9.tR_sHiA4owaHEPI6  
ecv5Na1CP6-  
ErkPgTRVqw1WVwuI
```

Decoded

```
HEADER:  
  
{  
  "alg": "HS256",  
  "typ": "JWT"  
}  
  
PAYLOAD:  
  
{  
  "id":  
  "65249840df3ad086f33bcc15",  
  "isAdmin": true,  
  "roles": [  
    {  
      "_id":  
      "651fb8cd54d4a06709b6117b",  
      "role": "Admin"  
    },  
    {  
      "_id":  
      "651fbac654d4a06709b6117e",  
      "role": "User"  
    }  
  ],  
  "iat": 1696904862  
}
```

Encoded

```
eyJhbGciOiJIUzI1NiIsIn  
R5cCI6IkpXVCJ9.eyJpZCI  
6IjY1MjQ5ODQwZGYzYWQwO  
DZmMzNiY2MxNSIsImlzQWR  
taW4iOnRydWUsInJvbGVzI  
jpbeJfaWQiOiiI2NTFmYjh  
jZDU0ZDRhMDY3MDliNjExN  
2IiLCJyb2xIjoiQWRtaW4  
ifSx7I19pZCI6IjY1MWZiY  
WM2NTRkNGEwNjcwOWI2MTE  
3ZSIIsInJvbGUiOijVc2Vyi  
n1dLCJpYXQiOjE2OTY5MDQ  
4NjJ9.tR_sHiA4owaHEPI6  
ecv5Na1CP6-  
ErkPgTRVqw1WVwuI
```

Decoded

```
HEADER:  
  
{  
  "alg": "HS256",  
  "typ": "JWT"  
}  
  
PAYLOAD:  
  
{  
  "id":  
  "65249840df3ad086f33bcc15",  
  "isAdmin": true,  
  "roles": [  
    {  
      "_id":  
      "651fb8cd54d4a06709b6117b",  
      "role": "Admin"  
    },  
    {  
      "_id":  
      "651fbac654d4a06709b6117e",  
      "role": "User"  
    }  
  ],  
  "iat": 1696904862  
}
```

The users' passwords have also been encrypted before storing on MongoDB.

```
export const register = async (req, res, next) => {

    const role = await Role.find({role: 'User'});
    const salt = await bcrypt.genSalt(10);
    const hashPassword = await bcrypt.hash(req.body.password, salt); // hash password
    const newUser = new User({
        firstName: req.body.firstName,
        lastName: req.body.lastName,
        username: req.body.userName,
        email: req.body.email,
        password: hashPassword, // password encryption using bcrypt
        roles: role
    });
    await newUser.save();
    return res.status(200).json("User Registration Successful");
}
```

User Registration Details Stored on MongoDB.

The screenshot shows the MongoDB Compass interface with the 'AuthDB' database selected. The 'users' collection is open, displaying a single document. The document structure is as follows:

```
_id: ObjectId('631f1bb0efffc07424fe0ed34e')
email: "johndoe@cloud.com"
firstName: "John"
lastName: "Doe"
password: "$2a$10$gkErQhvzHnareG02vdoU/CfJ9oiaAw22gp13bgdGeH5je"
profileImage: ""
roleId: 1
roles: [1]
createdAt: 2023-10-05T08:00:00Z
updatedAt: 2023-10-05T08:00:00Z
..._v: 0
```

Below the document, there is another partial document entry:

```
_id: ObjectId('631f1bb0efffc07239e837f4e0ed34e')
firstName: "Pankaj"
lastName: "Patel"
username: "pankajpatel"
email: "pankajpatel@cloud.com"
password: "$2a$10$gkErQhvzHnareG02vdoU/CfJ9oiaAw22gp13bgdGeH5je"
profileImage: ""
roleId: 2
roles: [2]
createdAt: 2023-10-05T08:00:00Z
updatedAt: 2023-10-05T08:00:00Z
..._v: 0
```

```

AuthDB.users
Documents Aggregations Schema Indexes Validation
Filter Type a query: { Field: "value" } or Generate query
8 1 DOCUMENTS INDEXES
1-8 of 8 Export Reset Find Options
_id: "001aef120c411f900efffc070424f5ed3e4"
email: "johndoe@cloud.com"
password: "$2b$12$uW9yj4i1327w4L3Ca5a5ad911apbhd411rrrH0HvA7Sjg."
profileImage: null
isAdmin: false
roles: ["user"]
createdAt: 2023-10-05T08:00:00.730Z
updatedAt: 2023-10-05T08:00:00.730Z
__v: 0

_id: "001aef120c411f900efffc070424f5ed3e5"
firstName: "John"
lastName: "Doe"
username: "johndoe"
email: "johndoe@cloud.com"
password: "$2b$12$uW9yj4i1327w4L3Ca5a5ad911apbhd411rrrH0HvA7Sjg."
profileImage: null
isAdmin: true
roles: ["admin"]
createdAt: 2023-10-05T08:00:00.730Z
updatedAt: 2023-10-05T08:00:00.730Z
__v: 0

```

Only the logged in Admins are allowed to view user info. And only the logged in user can view profile info filtered by ID.

```

authapp > api > routes > user.js > ...
1 import express from 'express';
2 export { getAllUsers, getByID } from '../controllers/user.controller.js';
3 import { verifyAdmin, verifyUser } from '../utils/verifyToken.js';
4
5 const router = express.Router();
6
7 // get all the users
8 router.get('/', verifyAdmin, getAllUsers); // only the admin is allowed to get all the users list
9
10 // get by id
11 router.get('/:id', verifyUser, getByID); // only the logged in user is allowed to view info
12
13 export default router;
14

```

User Registration Form with Validations

The screenshot shows a registration form on a web page. The form has several input fields: FirstName, LastName, Email, and Password. Each field has an associated error message indicating it is required. The 'Register' button at the bottom is disabled because none of the fields have been filled.

```

<div class="lg-w-3/2 mt-2 p-10 shadow-lg mx-auto">
  <h3 class="text-3xl text-center font-semibold py-5">Register</h3>
  <form [formGroup]="registerForm" class="flex flex-col"> <!--flex-col has been used as the input fields must be wrapped in a form group-->
    <div class="flex flex-col mt-4 mb-1">
      <input FormControlName="firstName" class="p-2 border border-gray-300 rounded" type="text" placeholder="First Name" />
      <span *ngIf="registerForm.hasError('required','firstName') && registerForm.controls['firstName'].dirty">First Name is required</span>
    </div>
    <div class="flex flex-col mt-4 mb-1">
      <input FormControlName="lastName" class="p-2 border border-gray-300 rounded" type="text" placeholder="Last Name" />
      <span *ngIf="registerForm.hasError('required','lastName') && registerForm.controls['lastName'].dirty">Last Name is required</span>
    </div>
    <div class="flex flex-col mt-4 mb-1">
      <input FormControlName="email" class="p-2 border border-gray-300 rounded" type="text" placeholder="Email" />
      <span *ngIf="registerForm.hasError('required','email') && registerForm.controls['email'].dirty">Email is required</span>
      <span *ngIf="registerForm.hasError('email','email') && registerForm.controls['email'].dirty">Email is invalid</span>
    </div>
    <div class="flex flex-col mt-4 mb-1">
      <input FormControlName="userName" class="p-2 border border-gray-300 rounded" type="text" placeholder="User Name" />
      <span *ngIf="registerForm.hasError('required','userName') && registerForm.controls['userName'].dirty">User Name is required</span>
    </div>
    <div class="flex flex-col mt-4 mb-1">
      <input FormControlName="password" class="p-2 border border-gray-300 rounded" type="password" placeholder="Password" />
      <span *ngIf="registerForm.hasError('required','password') && registerForm.controls['password'].dirty">Password is required</span>
    </div>
    <div class="flex flex-col mt-4 mb-1">
      <input FormControlName="confirmPassword" class="p-2 border border-gray-300 rounded" type="password" placeholder="Confirm Password" />
      <span *ngIf="registerForm.hasError('required','confirmPassword') && registerForm.controls['confirmPassword'].dirty">Confirm Password is required</span>
      <span *ngIf="registerForm.hasError('confirmPasswordValidator','confirmPassword') && registerForm.controls['confirmPassword'].dirty">Confirm Password is invalid</span>
    </div>
    <!--disabling the register button-->
    <button [disabled]="registerForm.invalid" (click)="register()" class="w-full py-2 px-4 bg-blue-500 text-white rounded border border-transparent transition-colors duration-300 ease-in-out">Register</button>

```

The screenshot shows the `register.component.ts` file. It defines a component named `RegisterComponent` that injects `FormBuilder` and `AuthService`. The component's `registerForm` property is a `FormGroup` containing four fields: `firstName`, `lastName`, `email`, and `password`. Each field has its own validation logic using `Validators.required` and `Validators.compose`.

```

import { Component } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { AuthService } from './services/auth.service';

@Component({
  selector: 'app-register',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule],
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.scss']
})

export default class RegisterComponent implements OnInit {
  registerForm!: FormGroup;

  constructor(private fb: FormBuilder, private authService: AuthService) {}

  ngOnInit(): void {
    this.registerForm = this.fb.group({
      firstName: ['', Validators.required],
      lastName: ['', Validators.required],
      email: ['', Validators.compose([Validators.required, Validators.email])], // email must be validate
      userName: ['', Validators.required],
      password: ['', Validators.required],
      confirmPassword: ['', Validators.required],
    }, {
      validator: confirmPasswordValidator('password', 'confirmPassword')
    })
  }
}

```

In the register form the 'Register' button has been disabled if the fields are empty.

All the required fields must be filled before the user can click on the register button.

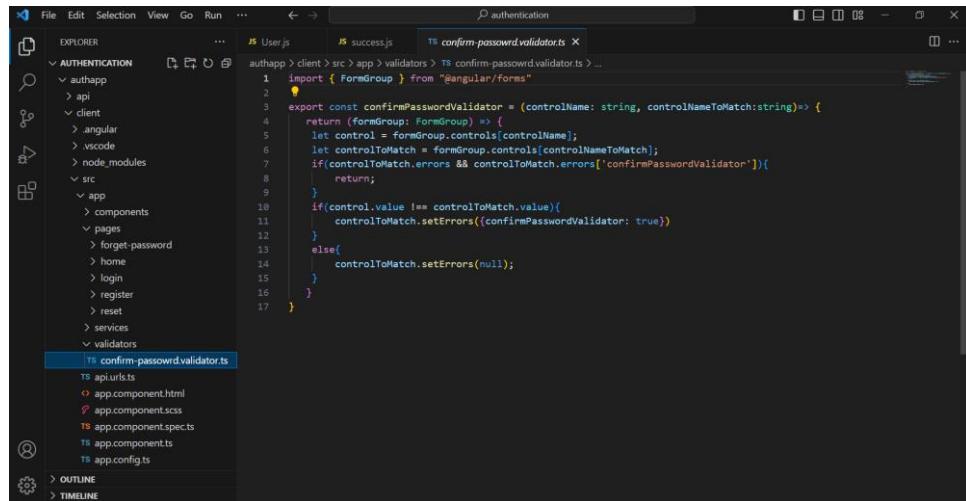
```
*ngIf="registerForm.hasError('required','firstName') &&
registerForm.controls['firstName'].dirty"
```

If the form fields become dirty, the required error will be displayed to the user. And only after all the required fields are filled with values, the submit button will be enabled and allowed to submit the registration details.

Creating custom validators for confirming password.

Validators folder of the application contains all the custom validators.

Ex: confirm-password.validator.ts file includes the custom validation for password confirmation.



The screenshot shows a code editor window with the title bar "authentication". The left sidebar is labeled "EXPLORER" and shows a project structure under "AUTENTICATION". The "validators" folder is expanded, and the file "confirm-password.validator.ts" is selected. The right pane displays the following TypeScript code:

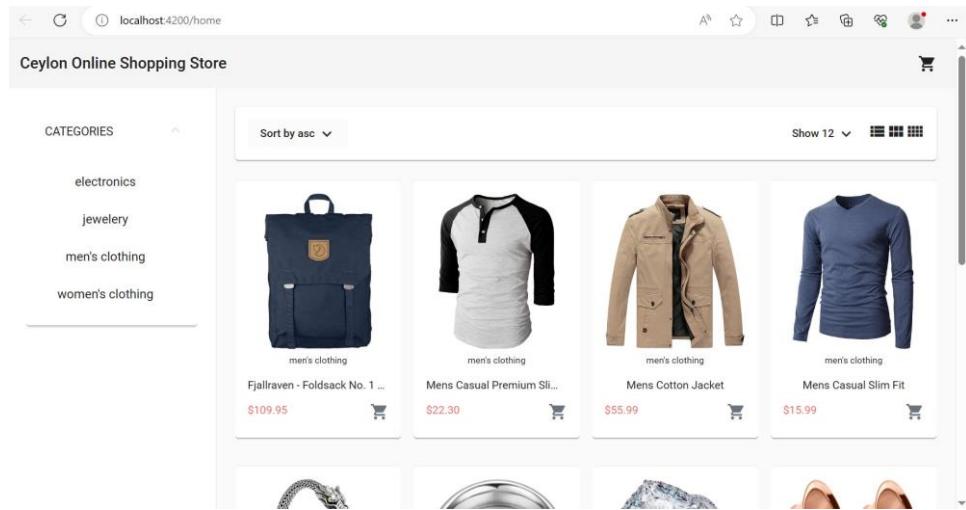
```
1 import { FormGroup } from "@angular/forms"
2
3 export const confirmPasswordValidator = (controlName: string, controlNameToMatch:string) => {
4   return (formGroup: FormGroup) => {
5     let control = formGroup.controls[controlName];
6     let controlToMatch = formGroup.controls[controlNameToMatch];
7     if(controlToMatch.errors && controlToMatch.errors['confirmPasswordValidator']){
8       return;
9     }
10    if(control.value !== controlToMatch.value){
11      controlToMatch.setErrors({confirmPasswordValidator: true})
12    }
13    else{
14      controlToMatch.setErrors(null);
15    }
16  }
17 }
```

An e commerce store

This application has been developed with the aim of providing customers with an exclusive online shopping experience. The application allows users to select a range of products also with the ability to categorize the products based on four different product categories. The product categories include electronics, jewellery, men's clothing and women's clothing.

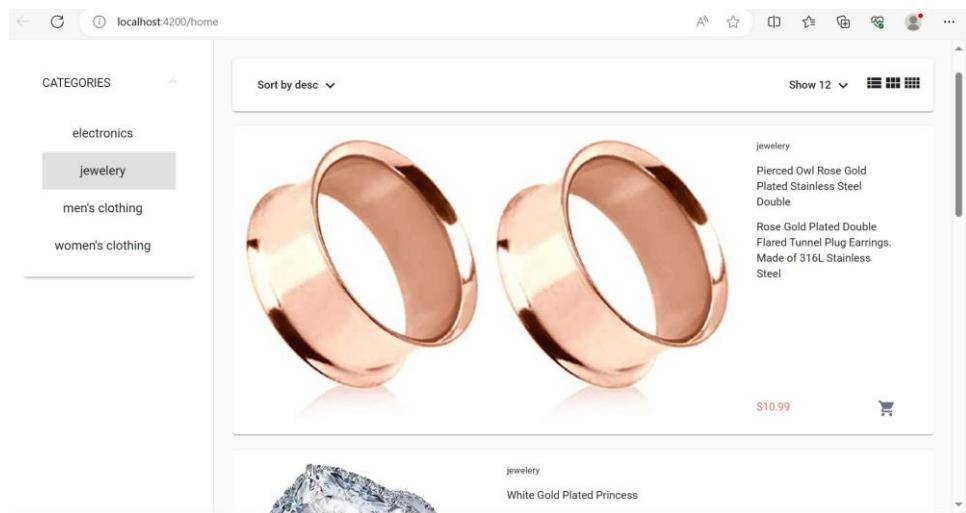
The categories section on the left side panel allows users to sort the products displayed on the shopping store. The header section on the top of the web store allows users to sort the displayed products according to the ascending order or descending order based on the user's preference.

On the right corner of this header section, the users can change the layout view of the products. Users can select how many items they wish to view along with the layout view preference.



Fake store API has been used here for displaying products and Stripe has been used for the payment integration.

Product Layout View 1



This allows users to view a single product on the web page at a time.

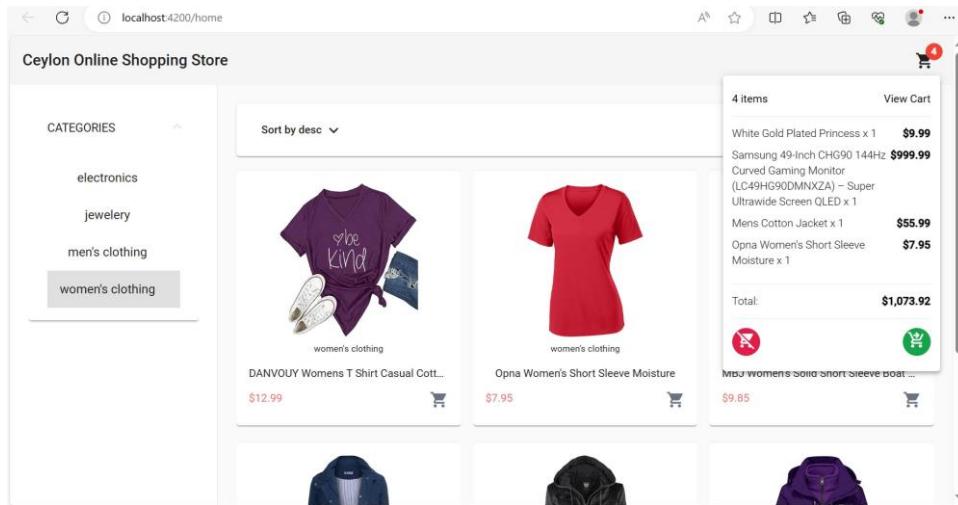
Product Layout View 2

This screenshot shows a product detail page for a Samsung QLED gaming monitor. The left sidebar lists categories: electronics, jewelery, men's clothing, and women's clothing, with 'electronics' selected. The main content area displays a large image of the monitor, which is curved and has the text 'QLED GAMING MONITOR' on its screen. To the right of the image is a detailed product description and a price of \$999.99.

Product Layout View 3

This screenshot shows a grid-based product listing for women's clothing. The left sidebar shows categories: electronics, jewelery, men's clothing, and women's clothing, with 'women's clothing' selected. The main content area displays three products: a purple t-shirt with 'be kind' text, a red t-shirt, and a white t-shirt. Each product has a small image, the brand name, a price (\$12.99, \$7.95, \$9.85), and a shopping cart icon.

Cart View 1



Products that have been added to the cart can be displayed by clicking on the cart item on the home page without navigating to the cart page.

The button displayed on the bottom left corner can be used to clear the products that have been already added to the cart.

The button on the bottom right corner here can be used to proceed to the cart page as displayed below.

Cart View 2

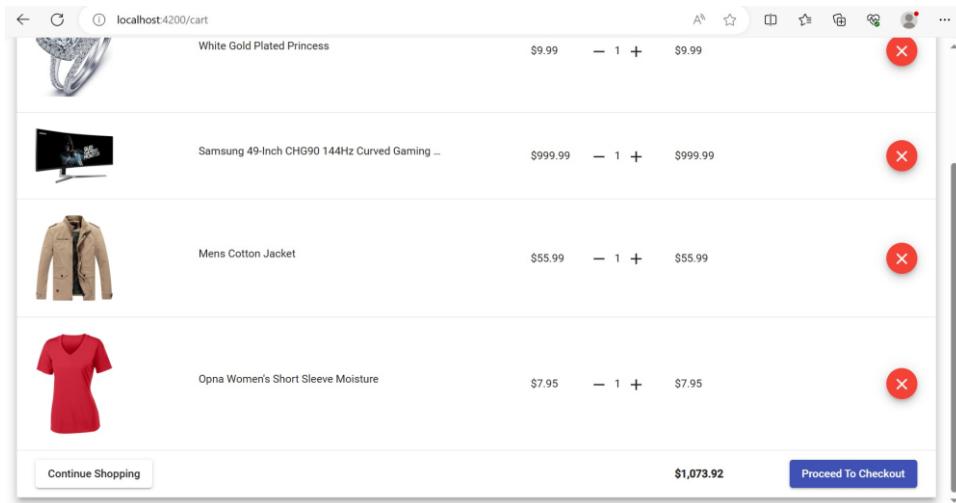
Product	Name	Price	Quantity	Total	
	White Gold Plated Princess	\$9.99	- 1 +	\$9.99	
	Samsung 49-Inch CHG90 144Hz Curved Gaming ...	\$999.99	- 1 +	\$999.99	
	Mens Cotton Jacket	\$55.99	- 1 +	\$55.99	
					

The “clear all” button displayed on the top right corner can be used to empty the cart by removing all the products added to the cart.

The quantity of the products can also be changed by clicking on the – and + signs.

Product	Name	Price	Quantity	Total	
	WD 4TB Gaming Drive Works with Playstation 4 P...	\$114.00	- 2 +	\$228.00	
	Pierced Owl Rose Gold Plated Stainless Steel Dou...	\$10.99	- 3 +	\$32.97	
	Mens Casual Slim Fit	\$15.99	- 1 +	\$15.99	
Continue Shopping				\$276.96	Proceed To Checkout

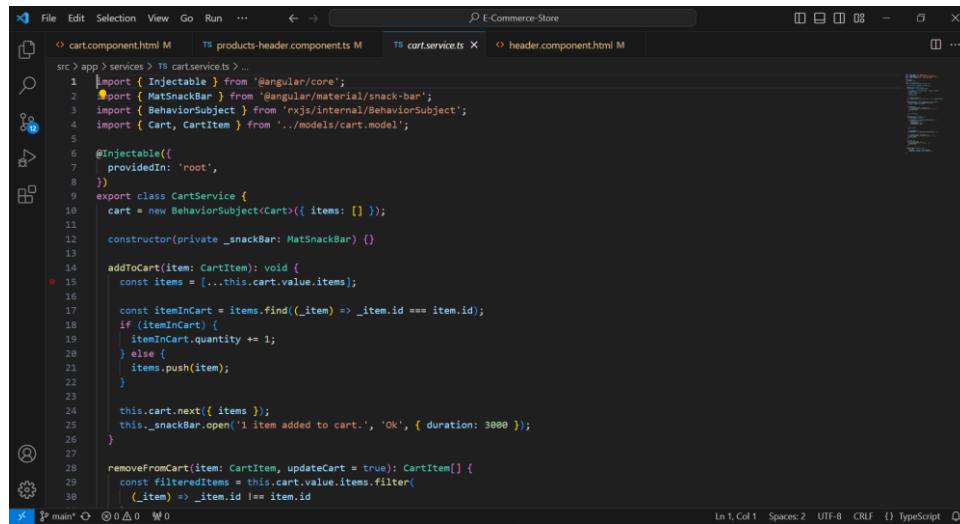
Cart View 3



"Continue Shopping" button displayed on the bottom left corner can be used to navigate back to the Home page and add more products to the cart.

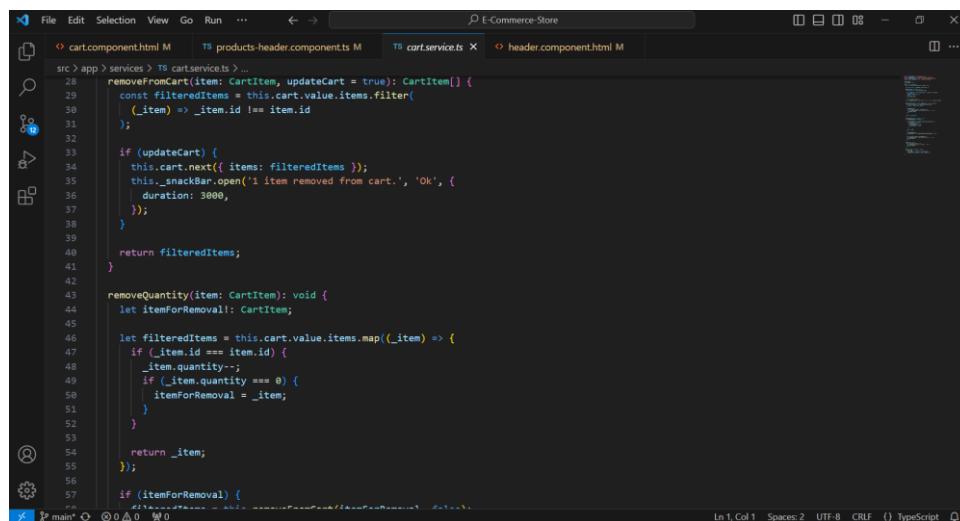
The total amount of all the products added to the cart can also be seen here on the bottom right corner of the cart page.

Cart Service



A screenshot of a code editor showing the `cartservice.ts` file. The code defines a `CartService` class with methods for adding items to the cart and removing them. It uses a `BehaviorSubject` to track the cart's state and a `MatSnackBar` for notifications.

```
src > app > services > cartservice.ts ...
1  import { Injectable } from '@angular/core';
2  import { MatSnackBar } from '@angular/material/snack-bar';
3  import { BehaviorSubject } from 'rxjs/internal/BehaviorSubject';
4  import { Cart, CartItem } from '../models/cart.model';
5
6  @Injectable({
7    providedIn: 'root',
8  })
9  export class CartService {
10    cart = new BehaviorSubject<Cart>({ items: [] });
11
12    constructor(private _snackBar: MatSnackBar) {}
13
14    addToCart(item: CartItem): void {
15      const items = [...this.cart.value.items];
16
17      const itemInCart = items.find((_item) => _item.id === item.id);
18      if (itemInCart) {
19        itemInCart.quantity += 1;
20      } else {
21        items.push(item);
22      }
23
24      this.cart.next({ items });
25      this._snackBar.open('1 item added to cart.', 'OK', { duration: 3000 });
26    }
27
28    removeFromCart(item: CartItem, updateCart = true): CartItem[] {
29      const filteredItems = this.cart.value.items.filter(
30        (_item) => _item.id !== item.id
31      );
32
33      if (updateCart) {
34        this.cart.next({ items: filteredItems });
35        this._snackBar.open('1 item removed from cart.', 'OK', { duration: 3000 });
36      }
37    }
38
39    return filteredItems;
40  }
41
42  removeQuantity(item: CartItem): void {
43    let itemForRemoval: CartItem;
44
45    let filteredItems = this.cart.value.items.map(_item => {
46      if (_item.id === item.id) {
47        _item.quantity--;
48        if (_item.quantity === 0) {
49          itemForRemoval = _item;
50        }
51      }
52
53      return _item;
54    });
55
56    if (itemForRemoval) {
57      // Logic to handle item removal
58    }
59  }
60
61  
```



A screenshot of a code editor showing the updated `cartservice.ts` file. The `removeFromCart` method now returns a new array of items, and the `removeQuantity` method has been added to handle reducing the quantity of an item in the cart.

```
src > app > services > cartservice.ts ...
28    removeFromCart(item: CartItem, updateCart = true): CartItem[] {
29      const filteredItems = this.cart.value.items.filter(
30        (_item) => _item.id !== item.id
31      );
32
33      if (updateCart) {
34        this.cart.next({ items: filteredItems });
35        this._snackBar.open('1 item removed from cart.', 'OK', { duration: 3000 });
36      }
37    }
38
39    return filteredItems;
40  }
41
42  removeQuantity(item: CartItem): void {
43    let itemForRemoval: CartItem;
44
45    let filteredItems = this.cart.value.items.map(_item => {
46      if (_item.id === item.id) {
47        _item.quantity--;
48        if (_item.quantity === 0) {
49          itemForRemoval = _item;
50        }
51      }
52
53      return _item;
54    });
55
56    if (itemForRemoval) {
57      // Logic to handle item removal
58    }
59  }
60
61  
```

A screenshot of the Visual Studio Code interface. The title bar says "E-Commerce-Store". The left sidebar shows a tree view with "src > app > services > cartservice.ts". The main editor tab is "cart.service.ts". The code is as follows:

```
src > app > services > cartservice.ts
54     itemForRemoval = _item;
55   }
56   }
57   return _item;
58 );
59 }
60
61 if (itemForRemoval) {
62   filteredItems = this.removeFromCart(itemForRemoval, false);
63 }
64
65 this.cart.next({ items: filteredItems });
66 this._snackBar.open('1 item removed from cart.', 'OK', {
67   duration: 3000,
68 });
69
70 }
71
72 }
73
74 getTotal(items: CartItem[]): number {
75   return items
76     .map(item => item.price * item.quantity)
77     .reduce((prev, current) => prev + current, 0);
78 }
79 }
```

The status bar at the bottom shows "Ln 1, Col 1 - Spaces: 2 - UTF-8 - CRLF - TypeScript".

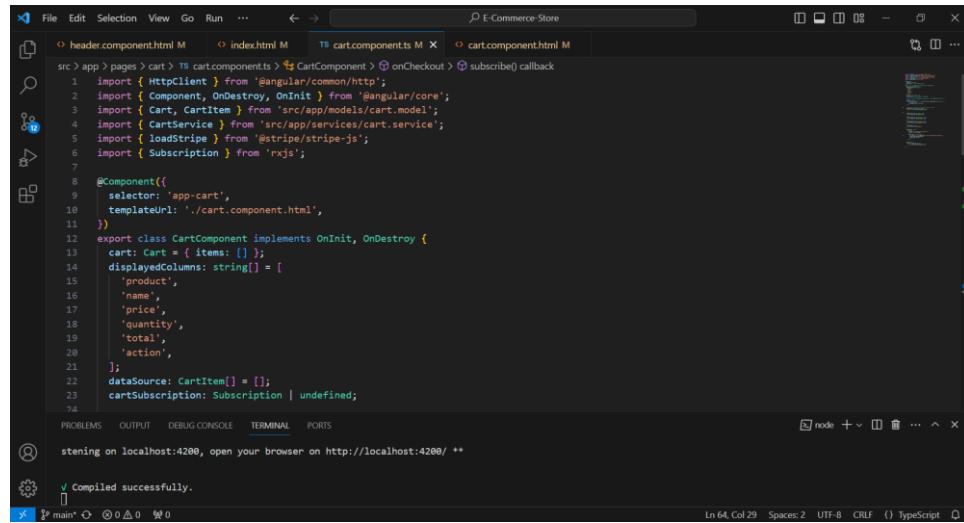
Adding API key for Stripe Payment Integration

A screenshot of the Visual Studio Code interface. The title bar says "E-Commerce-Store". The left sidebar shows a tree view with "src > app > pages > cart > cart.component.ts". The main editor tab is "cart.component.ts". The code is as follows:

```
src > app > pages > cart > cart.component.ts
60   .subscribe(async (res: any) => {
61     let stripe = await loadStripe('pk_test_7wktZBsd6whdiVcpfusNjIygt');
62     stripe.redirectToCheckout({
63       sessionId: res.id
64     });
65   });
66 }
67 }
68
69 ngOnDestroy() {
70   if (this.cartSubscription) {
71     this.cartSubscription.unsubscribe();
72   }
73 }
74 }
```

The status bar at the bottom shows "Ln 64, Col 29 - Spaces: 2 - UTF-8 - CRLF - TypeScript".

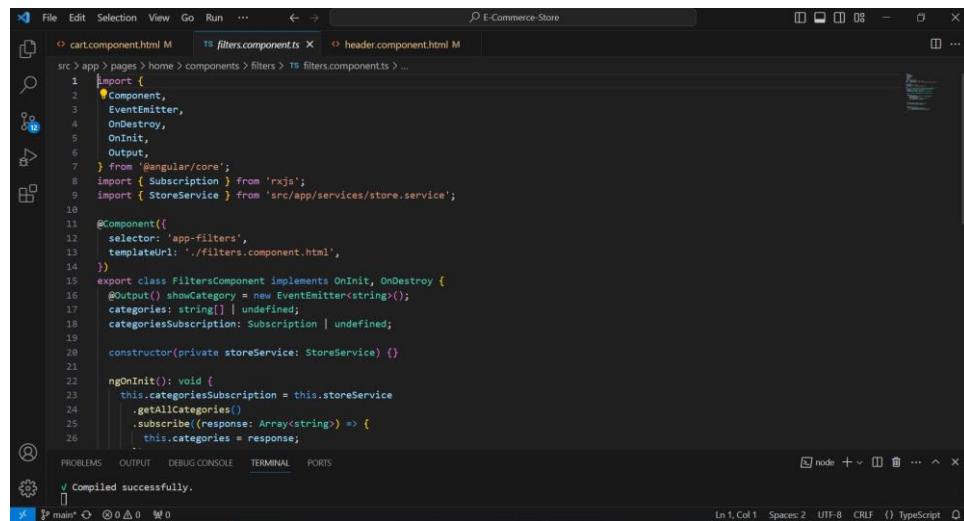
Cart Component Code



The screenshot shows the VS Code interface with the file `cart.component.ts` open. The code implements the `OnInit` and `OnDestroy` lifecycle hooks. It defines a `cart` object with items and displayed columns, and a `dataSource` of type `CartItem[]`. It also handles a `cartSubscription` for updates. The component selector is `'app-cart'`.

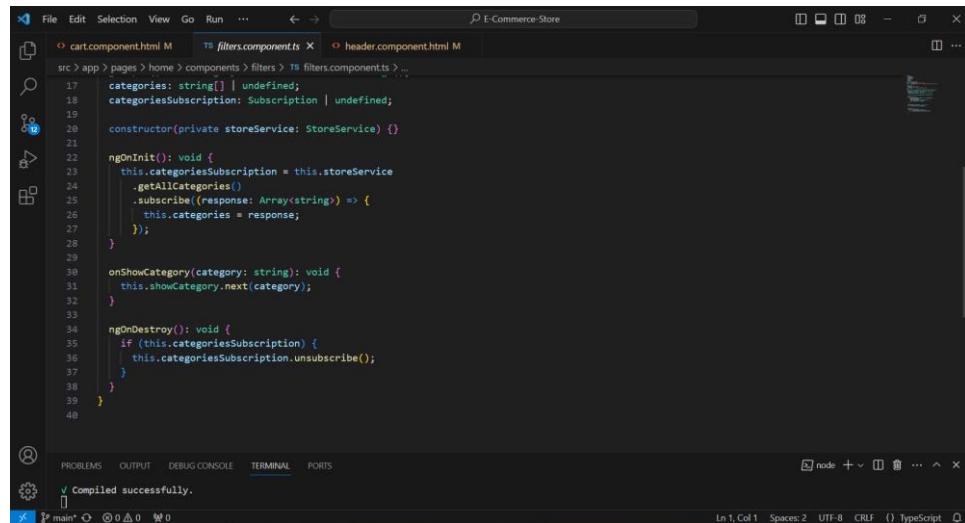
```
src > app > pages > cart > cart.component.ts M | cart.component.html M | cart.component.ts M X | cart.component.html M
1 import { HttpClient } from '@angular/common/http';
2 import { Component, OnDestroy, OnInit } from '@angular/core';
3 import { Cart, CartItem } from 'src/app/models/cart.model';
4 import { CartService } from 'src/app/services/cart.service';
5 import { loadStripe } from '@stripe/stripe-js';
6 import { Subscription } from 'rxjs';
7
8 @Component({
9   selector: 'app-cart',
10  templateUrl: './cart.component.html',
11 })
12 export class CartComponent implements OnInit, OnDestroy {
13   cart: Cart = { items: [] };
14   displayedColumns: string[] = [
15     'product',
16     'name',
17     'price',
18     'quantity',
19     'total',
20     'action',
21   ];
22   dataSource: CartItem[] = [];
23   cartSubscription: Subscription | undefined;
24
25   constructor(private cartService: CartService) {}
26
27   ngOnInit(): void {
28     this.cartSubscription = this.cartService.getCart();
29     this.cartSubscription?.subscribe((cart) => {
30       this.cart = cart;
31     });
32   }
33
34   ngOnDestroy(): void {
35     this.cartSubscription?.unsubscribe();
36   }
37
38   onCheckout(): void {
39     const stripePromise = loadStripe('pk_test_5IjXrJGKQHgkWzqf');
40     stripePromise
41       .then((stripe) => {
42         stripe.redirectToCheckout({ sessionId: this.cart.id });
43       })
44       .catch((error) => {
45         console.error(error);
46       });
47   }
48
49   onAdd(item: CartItem): void {
50     this.cart.items.push(item);
51   }
52
53   onRemove(item: CartItem): void {
54     this.cart.items = this.cart.items.filter((i) => i !== item);
55   }
56
57   onQuantityChange(item: CartItem, quantity: number): void {
58     this.cart.items = this.cart.items.map((i) => {
59       if (i === item) {
60         i.quantity = quantity;
61       }
62       return i;
63     });
64   }
65
66   onTotalChange(): void {
67     this.cart.total = this.cart.items.reduce((total, item) => {
68       return total + item.price * item.quantity;
69     }, 0);
70   }
71
72   onAction(item: CartItem, action: string): void {
73     switch (action) {
74       case 'remove':
75         this.onRemove(item);
76         break;
77       case 'update':
78         this.onQuantityChange(item, item.quantity + 1);
79         break;
80       case 'decrease':
81         this.onQuantityChange(item, item.quantity - 1);
82         break;
83     }
84   }
85
86   onCartUpdated(cart: Cart): void {
87     this.cart = cart;
88   }
89
90   onCartSubscriptionError(error: any): void {
91     console.error(error);
92   }
93 }
```

Filters Component



The screenshot shows the VS Code interface with the file `filters.component.ts` open. The code implements the `OnInit` lifecycle hook. It initializes an `Output` event emitter for categories, subscribes to the `getStoreCategories` service, and updates the `categories` array with the received response.

```
src > app > pages > home > components > filters > filters.component.ts > ...
1 import {
2   Component,
3   EventEmitter,
4   OnDestroy,
5   OnInit,
6   Output,
7 } from '@angular/core';
8 import { Subscription } from 'rxjs';
9 import { StoreService } from 'src/app/services/store.service';
10
11 @Component({
12   selector: 'app-filters',
13   templateUrl: './filters.component.html',
14 })
15 export class FiltersComponent implements OnInit, OnDestroy {
16   @Output() showCategory = new EventEmitter<string>();
17   categories: string[] | undefined;
18   categoriesSubscription: Subscription | undefined;
19
20   constructor(private storeService: StoreService) {}
21
22   ngOnInit(): void {
23     this.categoriesSubscription = this.storeService
24       .getCategories()
25       .subscribe((response: Array<string>) => {
26         this.categories = response;
27       });
28   }
29
30   ngOnDestroy(): void {
31     this.categoriesSubscription?.unsubscribe();
32   }
33 }
```



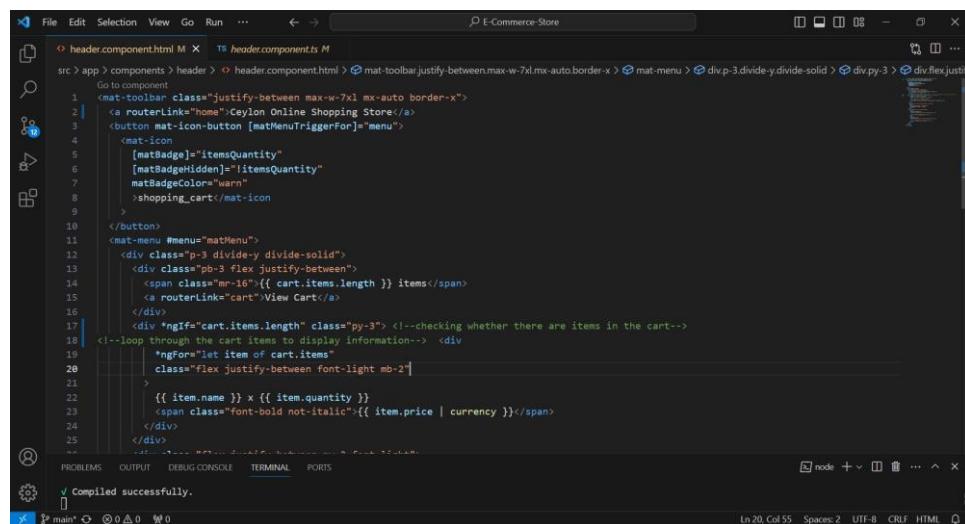
```
src > app > pages > home > components > filters > filters.component.ts > ...
17 categories: string[] | undefined;
18 categoriesSubscription: Subscription | undefined;
19
20 constructor(private storeService: StoreService) {}
21
22 ngOnInit(): void {
23   this.categoriesSubscription = this.storeService
24     .getAllCategories()
25     .subscribe((response: Array<string>) => {
26       this.categories = response;
27     });
28 }
29
30 onShowCategory(category: string): void {
31   this.showCategory.next(category);
32 }
33
34 ngOnDestroy(): void {
35   if (this.categoriesSubscription) {
36     this.categoriesSubscription.unsubscribe();
37   }
38 }
39 }
40
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Compiled successfully.

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF () TypeScript

Header Component



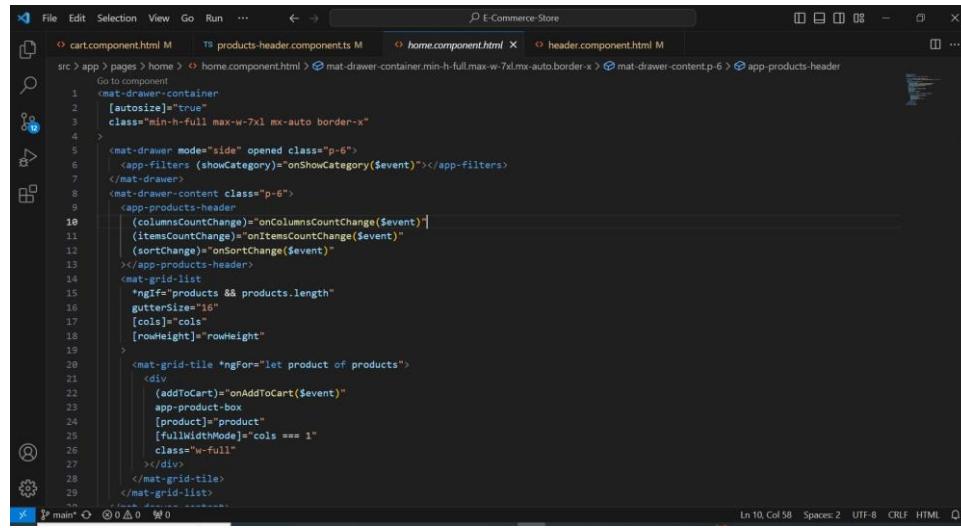
```
src > app > components > header > header.component.ts > ...
Go to component
1 <mat-toolbar class="justify-between max-w-7xl mx-auto border-x">
2   <a routerLink="home">Ceylon Online Shopping Store</a>
3   <button mat-icon-button [matMenuTriggerFor]="menu">
4     <mat-icon
5       [matBadge]="itemsQuantity"
6       [matBadgeHidden]="itemsQuantity"
7       matBadgeColor="warn"
8       >shopping_cart</mat-icon>
9   </button>
10  <mat-menu #menu="matMenu">
11    <div class="p-3 divide-y divide-solid">
12      <div class="pb-3 flex justify-between">
13        <span class="mr-16">{{ cart.items.length }} items</span>
14        <a routerLink="cart">View Cart</a>
15      </div>
16      <div *ngIf="cart.items.length" class="py-3"> <!---checking whether there are items in the cart-->
17        <!---loop through the cart items to display information--> <div
18          ngFor="let item of cart.items"
19          class="flex justify-between font-light mb-2">
20            <div> {{ item.name }} x {{ item.quantity }} </div>
21            <span class="font-bold not-italic">{{ item.price | currency }}</span>
22          </div>
23        </div>
24      </div>
25    </div>
26  </mat-menu>
27</mat-toolbar>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Compiled successfully.

Ln 20, Col 55 Spaces: 2 UTF-8 CRLF HTML

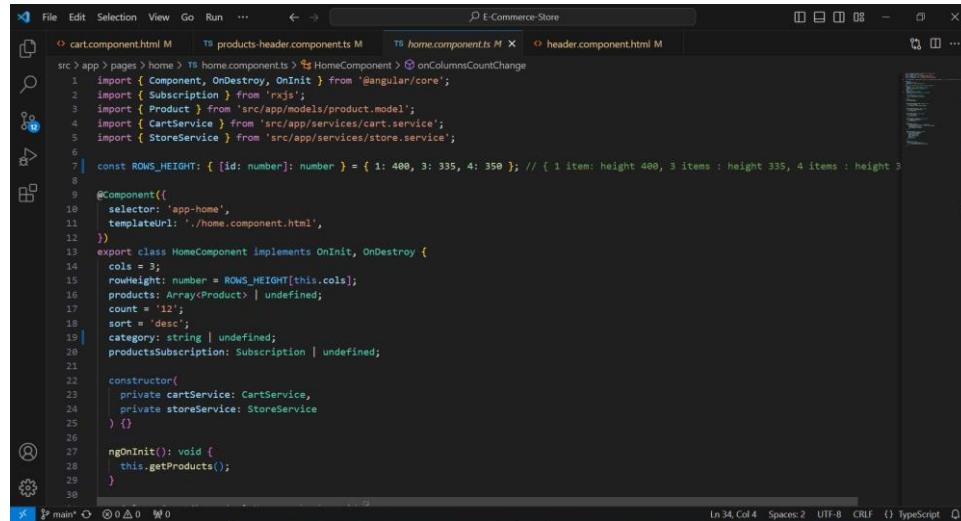
Home Component



The screenshot shows the `home.component.html` file in a code editor. The template uses the `mat-grid-list` component to layout child components into columns and rows. The code includes event handlers for column count change, item count change, and sort change.

```
<mat-grid-list *ngIf="products && products.length" gutterSize="16" [cols]="cols" [rowHeight]="rowHeight">
  <mat-grid-tile *ngFor="let product of products">
    <div (addToCart)="onAddToCart($event)">
      <app-product-box [product]="product" [fullWidthMode]="cols === 1" class="w-full">
        ...
      </app-product-box>
    </div>
  </mat-grid-tile>
</mat-grid-list>
```

Here, `mat-grid-list` separates the child components into columns and rows.



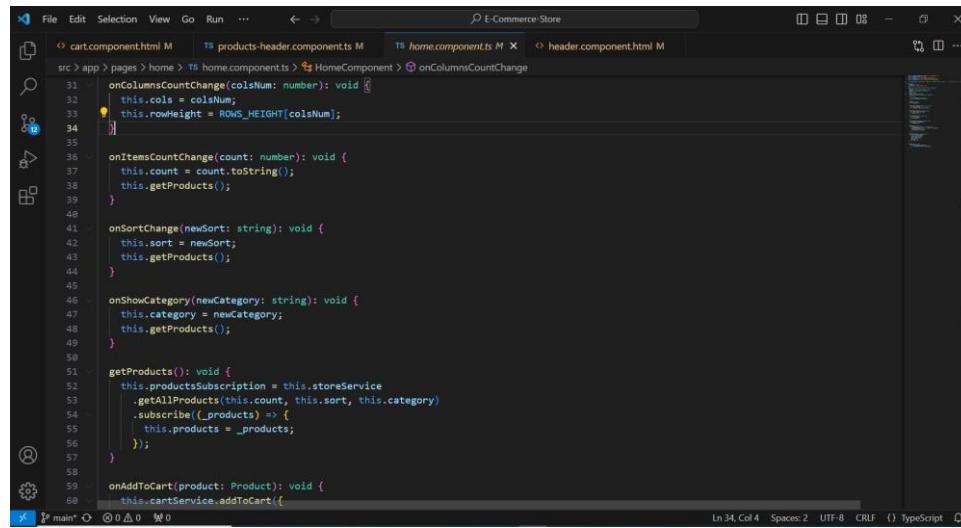
The screenshot shows the `home.component.ts` file in a code editor. It defines a `ROWS_HEIGHT` constant and an `HomeComponent` class that implements `OnInit` and `OnDestroy`. The component has properties for `cols`, `rowHeight`, `products`, `count`, `sort`, `category`, and `productsSubscription`. It also has a constructor for `CartService` and `StoreService`, and an `ngOnInit` method that calls `getProducts`.

```
const ROWS_HEIGHT: { [id: number]: number } = { 1: 400, 3: 335, 4: 350 }; // { 1 item: height 400, 3 items : height 335, 4 items : height 350 }

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
})
export class HomeComponent implements OnInit, OnDestroy {
  cols = 3;
  rowHeight: number = ROWS_HEIGHT[this.cols];
  products: Array<Product> | undefined;
  count = '12';
  sort = 'desc';
  category: string | undefined;
  productsSubscription: Subscription | undefined;

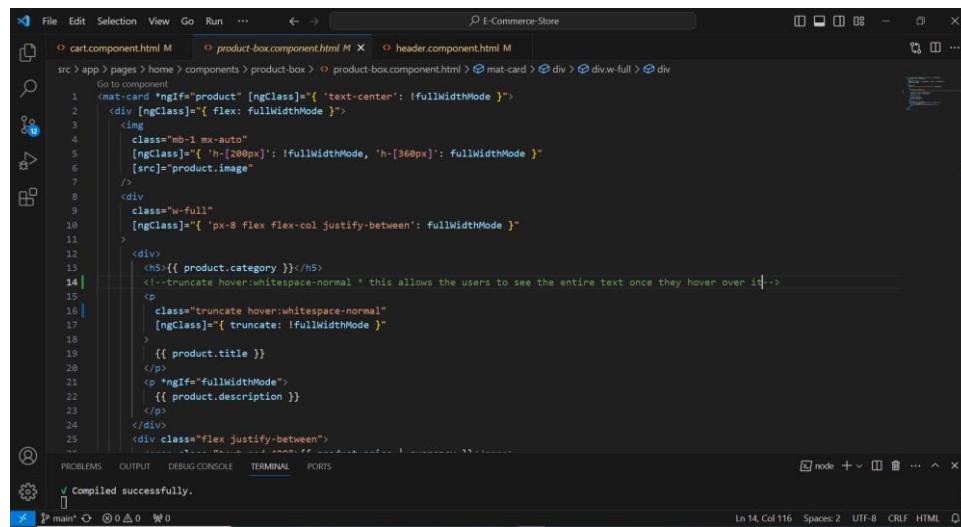
  constructor(
    private cartService: CartService,
    private storeService: StoreService
  ) {}

  ngOnInit(): void {
    this.getProducts();
  }
}
```



```
src > app > pages > home > TS home.component.ts M TS home.component.ts M onColumnsCountChange
31 |     onColumnsCountChange(colsNum: number): void {
32 |         this.cols = colsNum;
33 |         this.rowHeight = ROWS_HEIGHT[colsNum];
34 |     }
35 |
36 |     onItemCountChange(count: number): void {
37 |         this.count = count.toString();
38 |         this.getProducts();
39 |     }
40 |
41 |     onSortChange(newSort: string): void {
42 |         this.sort = newSort;
43 |         this.getProducts();
44 |     }
45 |
46 |     onShowCategory(newCategory: string): void {
47 |         this.category = newCategory;
48 |         this.getProducts();
49 |     }
50 |
51 |     getProducts(): void {
52 |         this.productsSubscription = this.storeService
53 |             .getAllProducts(this.count, this.sort, this.category)
54 |             .subscribe(_products => {
55 |                 this.products = _products;
56 |             });
57 |     }
58 |
59 |     onAddToCart(product: Product): void {
60 |         this.cartService.addToCart({
61 |             id: product.id,
62 |             title: product.title,
63 |             description: product.description,
64 |             price: product.price,
65 |             image: product.image
66 |         });
67 |     }
68 | }
```

Product Box Component



```
src > app > pages > home > components > product-box > TS product-box.component.html M onColumnsCountChange
1 <mat-card *ngIf="product" [ngClass]="{ 'text-center': !fullWidthMode }>
2     <div [ngClass]="'flex: fullWidthMode'>
3         <img
4             class="mb-1 mx-auto"
5             [ngClass]="'h-[280px]': fullWidthMode, 'h-[360px]': fullWidthMode "
6             [src]=product.image
7         />
8         <div
9             class="w-full"
10            [ngClass]="'px-8 flex flex-col justify-between': fullWidthMode "
11        >
12            <div>
13                <h3>{{ product.category }}</h3>
14                <!--truncate hover::whitespace-normal * this allows the users to see the entire text once they hover over it-->
15                <p
16                    class="truncate hover::whitespace-normal"
17                    [ngClass]="' truncate: !fullWidthMode '"
18                >
19                    {{ product.title }}
20                </p>
21                <p *ngIf="fullWidthMode">
22                    {{ product.description }}
23                </p>
24            </div>
25            <div class="flex justify-between">
```

Once the user hover over a specific item, the entire title description for that product can be viewed by the users.

The screenshot shows the VS Code interface with the product-box.component.ts file open. The code defines a component for a product box, including a template URL and a class named ProductBoxComponent. It includes properties for full width mode and a product, and an event emitter for adding to the cart. The terminal shows a successful build and compilation.

```
src > app > pages > home > components > product-box > TS product-box.components > ProductBoxComponent
1 import { Component, EventEmitter, Input, Output } from '@angular/core';
2 import { Product } from 'src/app/models/product.model';
3
4 @Component({
5   selector: '[app-product-box]',
6   templateUrl: './product-box.component.html',
7 })
8 export class ProductBoxComponent {
9   @Input() fullWidthMode = false; // full width mode has been disabled at first
10  @Input() product: Product | undefined;
11  @Output() addToCart = new EventEmitter();
12
13  constructor() {}
14
15  onAddToCart(): void {
16    this.addToCart.emit(this.product);
17  }
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
4.81 MB
Build at: 2023-11-22T12:57:37.077Z - Has
h: 89a34f92ed28121 - Time: 471ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
Compiled successfully.
Ln 9, Col 83 Spaces: 2 CRLF: ⚡ TypeScript: Q

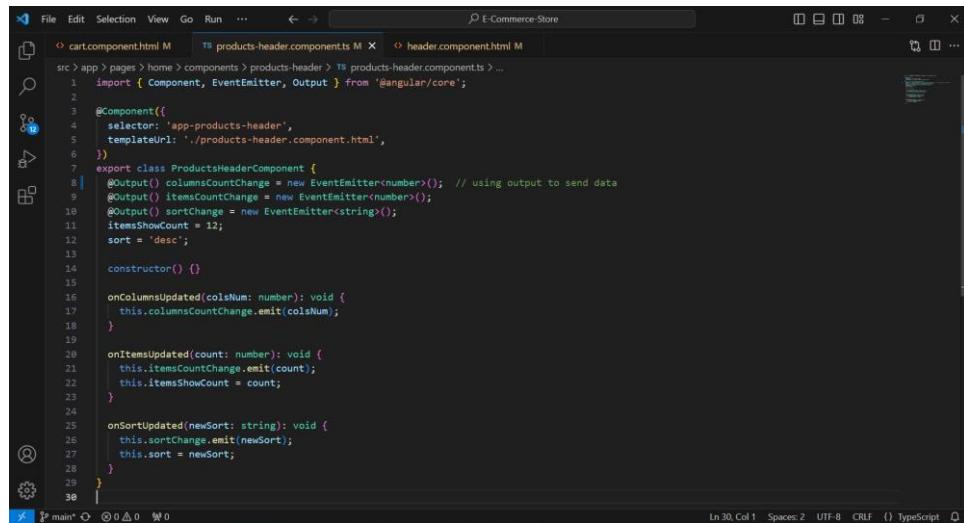
```

Product Header Component

The screenshot shows the VS Code interface with the products-header.component.html file open. The code defines a mat-card component with a flex justify-between layout. It contains a button for sorting by menu, a mat-menu with sort options (desc and asc), and a button for showing items. The terminal shows a successful build and compilation.

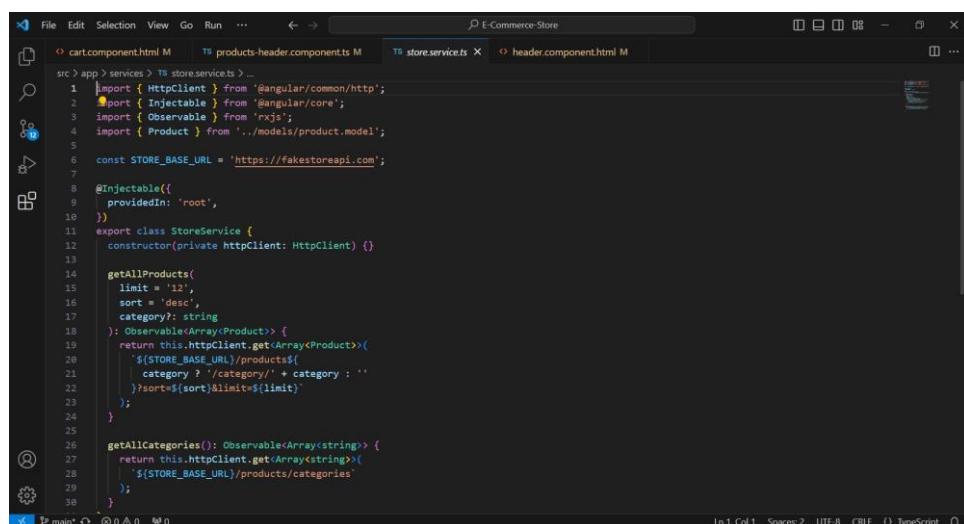
```
src > app > pages > home > components > products-header > products-header.component.html > header.component.html > mat-card.mb-4
1 <mat-card class="mb-4">
2   <div class="flex justify-between">
3     <div>
4       <button mat-button [matMenuTriggerFor]="sortByMenu">
5         Sort by {{ sort }}
6         <mat-icon>expand_more</mat-icon>
7       </button>
8       <mat-menu #sortByMenu="matMenu">
9         <button (click)="onSortUpdated('desc')">mat-menu-item:desc</button>
10        <button (click)="onSortUpdated('asc')">mat-menu-item:asc</button>
11      </mat-menu>
12    </div>
13    <div class="flex items-center">
14      <div>
15        <button mat-button [matMenuTriggerFor]="menu">
16          Show {{ itemsShowCount }}
17        </button>
18      </div>
19    </div>
20  </div>
21</mat-card>
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
4.81 MB
Build at: 2023-11-22T12:57:37.077Z - Has
h: 89a34f92ed28121 - Time: 471ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
Compiled successfully.
Ln 1, Col 1 Spaces: 2 CRLF: ⚡ HTML: Q

```



```
src > app > pages > home > components > products-header > products-header.component.ts ...  
1 import { Component, EventEmitter, Output } from '@angular/core';  
2  
3 @Component({  
4   selector: 'app-products-header',  
5   templateUrl: './products-header.component.html',  
6 })  
7 export class ProductsHeaderComponent {  
8   @Output() columnsCountChange = new EventEmitter<number>(); // Using output to send data  
9   @Output() itemsCountChange = new EventEmitter<number>();  
10  @Output() sortChange = new EventEmitter<string>();  
11  itemsShowCount = 12;  
12  sort = 'desc';  
13  
14  constructor() {}  
15  
16  onColumnsUpdated(colsNum: number): void {  
17    this.columnsCountChange.emit(colsNum);  
18  }  
19  
20  onItemsUpdated(count: number): void {  
21    this.itemsCountChange.emit(count);  
22    this.itemsShowCount = count;  
23  }  
24  
25  onSortUpdated(newSort: string): void {  
26    this.sortChange.emit(newSort);  
27    this.sort = newSort;  
28  }  
29}  
30
```

Store Service code



```
src > app > services > store.service.ts ...  
1 import { HttpClient } from '@angular/common/http';  
2 import { Injectable } from '@angular/core';  
3 import { Observable } from 'rxjs';  
4 import { Product } from '../models/product.model';  
5  
6 const STORE_BASE_URL = 'https://fakestoreapi.com';  
7  
8 @Injectable({  
9   providedIn: 'root',  
10 })  
11 export class StoreService {  
12   constructor(private httpClient: HttpClient) {}  
13  
14   getAllProducts(  
15     limit = '12',  
16     sort = 'desc',  
17     category?: string  
18   ): Observable<Array<Product>> {  
19     return this.httpClient.get<Array<Product>>(`  
20       ${STORE_BASE_URL}/products${  
21         category ? `/category/${category}` : ''  
22       }?sort=${sort}&limit=${limit}`  
23     );  
24   }  
25  
26   getAllCategories(): Observable<Array<string>> {  
27     return this.httpClient.get<Array<string>>(`  
28       ${STORE_BASE_URL}/products/categories`  
29     );  
30   }  
31 }
```

Fake Store API has been used to display products on the web store.

Dear Sir/Madam,

I have not been able to fully complete this coursework as a result of my severe health condition which had been affecting my day-to-day activities in a negative manner over a long period of time. I have also been attending an internship in Japan at a Japanese IT company which also affected my ability to work on this.

I sincerely apologise for not being able to work my best on this coursework.

This is the only coursework that I had to complete as a repeat module during the last three years of my degree program. I have been able to pass all the exams and course works at the first attempt and I was keen on maintaining that perfect academic record until the end of my degree program. However, my health condition and some other factors have collectively affected my ability work longer periods of time and perform at my full capacity. The pain associated with my health condition sometimes last for couple of hours which leaves me incapable of working as I have to lay down after applying and taking the medications.

I kindly request you to allow me to grant a passing mark as I may not get another opportunity to complete this coursework before graduation.

I have also included some documents as evidence.

Thank You!

RUHIRU MEDI CARE CENTRE

Dr T I Weerasinghe
MBBS (Peradeniya)(SL)
Reg No. 20268
HWH

2/2, Galwana Junction,
Udumulla road,
Mulleriyawa, Sri Lanka
Tel: 0767628545

To: Dean, School of Engineering, Computing, Mathematics

Medical Certificate

I certify that Mr. P.A.S Prasad was under my care since January, receiving treatments for a chronic anal fissure and its associated symptoms of rectal bleeding, constipation, and severe abdominal pain during and after bowel movements. Mr. Prasad's condition has not been improved with medicine and other treatments and several recurrences of the condition has occurred during the last couple of months.

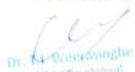
As a result, I have determined that he needs to go through a Lateral Internal Sphincterotomy surgery as the next immediate step of treatment. I have further instructed Mr. Prasad to consult Dr Sanjeewa Seneviratne (Consultant Surgeon) to receive further treatments including the surgery.

I hereby confirm that Mr. Prasad was not in good condition to carry out academic studies and other related tasks during the several recurrences of his condition which affected the overall quality of his day to day activities.

Additionally, it may take 1-2 weeks for him to recover after the surgery and get back to his normal routine and I request to provide him with the assistance he needs throughout this time.

Date: 24/03/2023

Medical Officer:


Dr. T.I. Weerasinghe
MBBS (Peradeniya)
Reg. No. 20268

29/03/2023

Internship Offer

Dear Palle Arachchilage Sachintha Prasad,

I am pleased to confirm your internship as "Sri Lanka Embedded Engineer Training Program" in BRYCEN Co. Ltd. Your duties and assignments for this position are as follows.

For the purpose of working as an advanced IT engineer in Japan in the future

- (1) Acquisition of industrial equipment development in Japan
- (2) Acquisition of teamwork in product development
- (3) Accustomed to the living environment in Japan

Your first day of work will be 1st August 2023, and your internship is scheduled to end on 31st October 2023, in 30F St.Luke's Tower, 8-1 Akashi-cho, Chuo-ku, Tokyo 104-6591 Japan.

We provide flight tickets, company housing, living expenses, etc.

If you have any questions, please feel free to contact Yuki Kitamura (ykitamura@brycen.co.jp). We wish you will have a good work experience to improve your career opportunities during the internship.

Sincerely,



Yuki Kitamura

 Human Resources Department

 Brycen Co., Ltd.

St.Luke's Tower 30th floor, 8-1 Akashicho Minato-Ku Tokyo

Phone: +81-(0)3- 6264-7221



本部：東京都 中央区 入船 1-5-11 弘報ビル 5 階 〒104-0042
1-5-11 Irifune Chuo Tokyo Japan 104-0042
TEL:03-6372-0211 FAX:03-6372-0212 Email : jasainfo@jasa.or.jp
支部：北海道、東北、関東、中部、北陸、近畿、九州