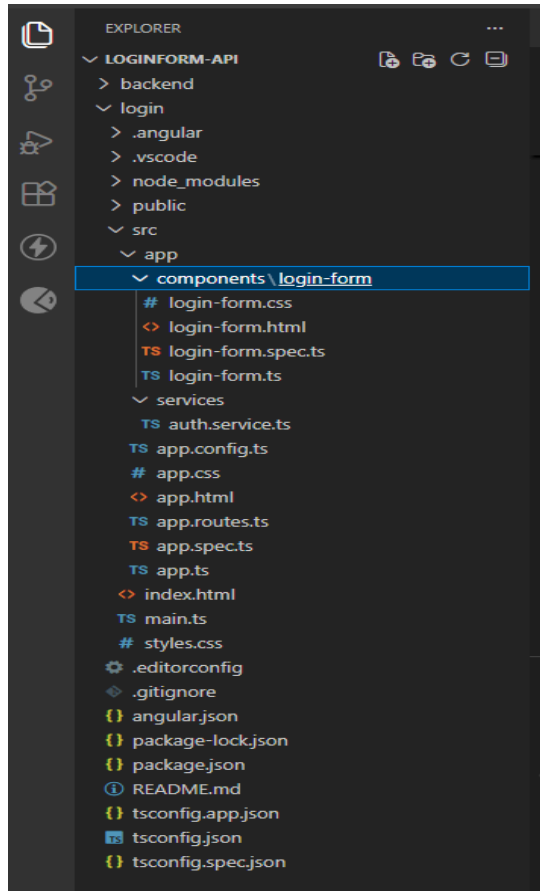


Practical No : 05

Aim : - Connecting Angular Login Form to rest API

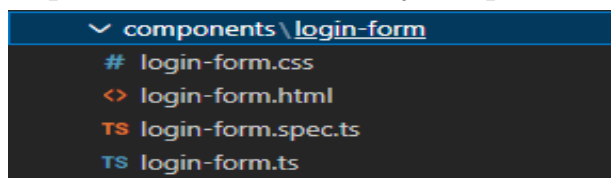
Step 1 : Creating a new file



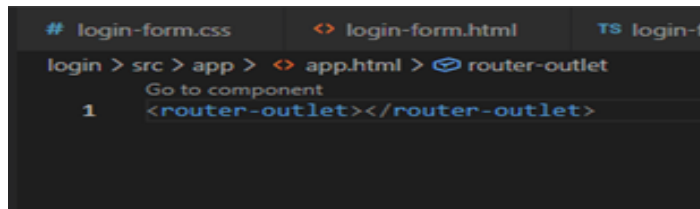
Step 2 : Open the terminal and run the required commands

```
C:\Users\Anish\Documents\MeanStack\login>ng g c components/login-form
Node.js version v25.2.1 detected.
Odd numbered Node.js versions will not enter LTS status and should not be
dejs.org/en/about/previous-releases/.
CREATE src/app/components/login-form/login-form.spec.ts (576 bytes)
CREATE src/app/components/login-form/login-form.ts (212 bytes)
CREATE src/app/components/login-form/login-form.css (0 bytes)
CREATE src/app/components/login-form/login-form.html (26 bytes)
```

Step 3 : Create the necessary components

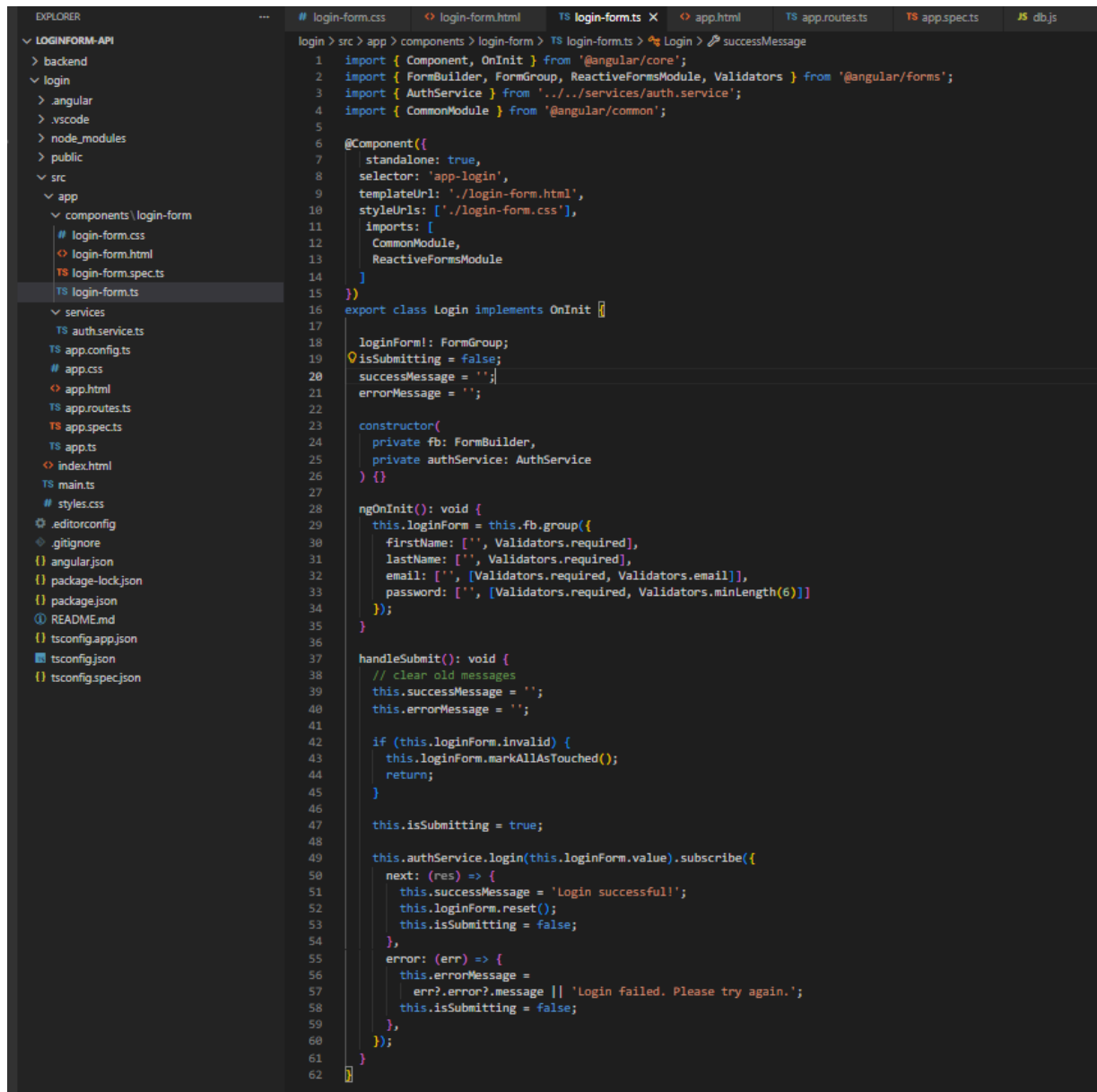


Step 4: Edit the app.html file to define the main structure



```
# login-form.css login-form.html TS login-form.ts
login > src > app > app.html > router-outlet
Go to component
1 <router-outlet></router-outlet>
```

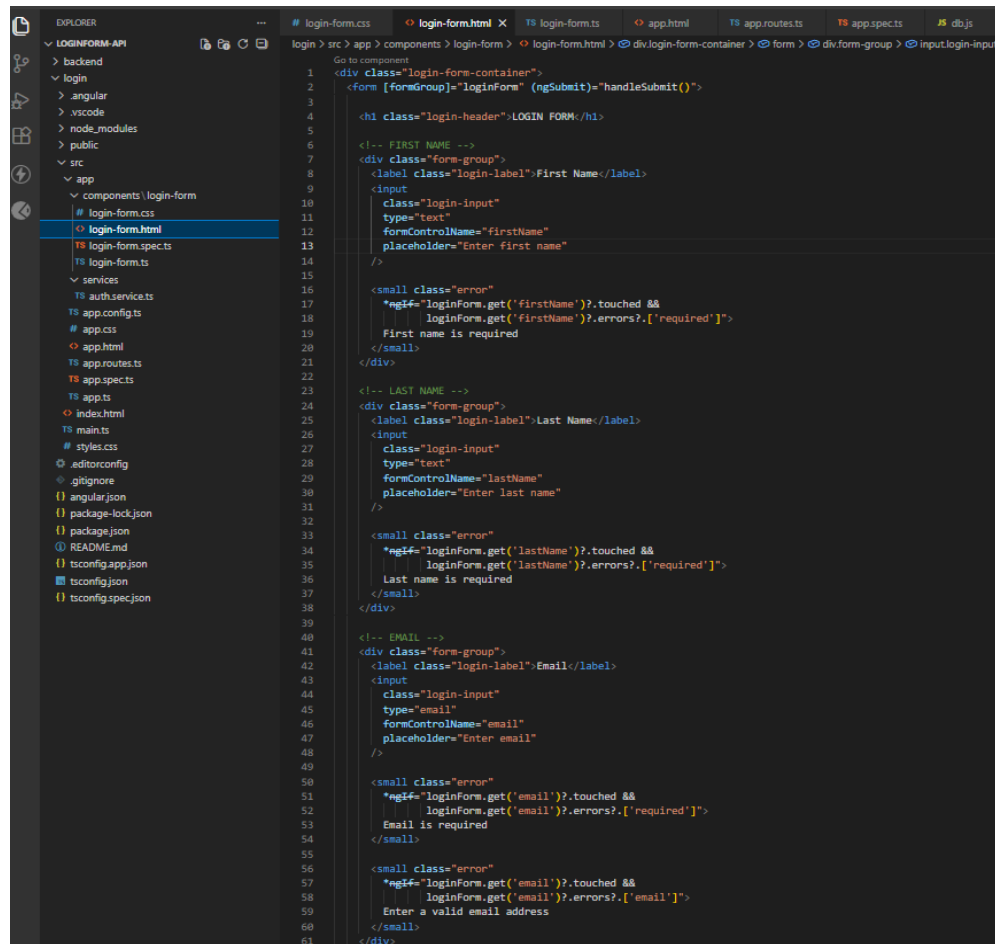
Step 5 : Edit the login-form.ts file to write the logic



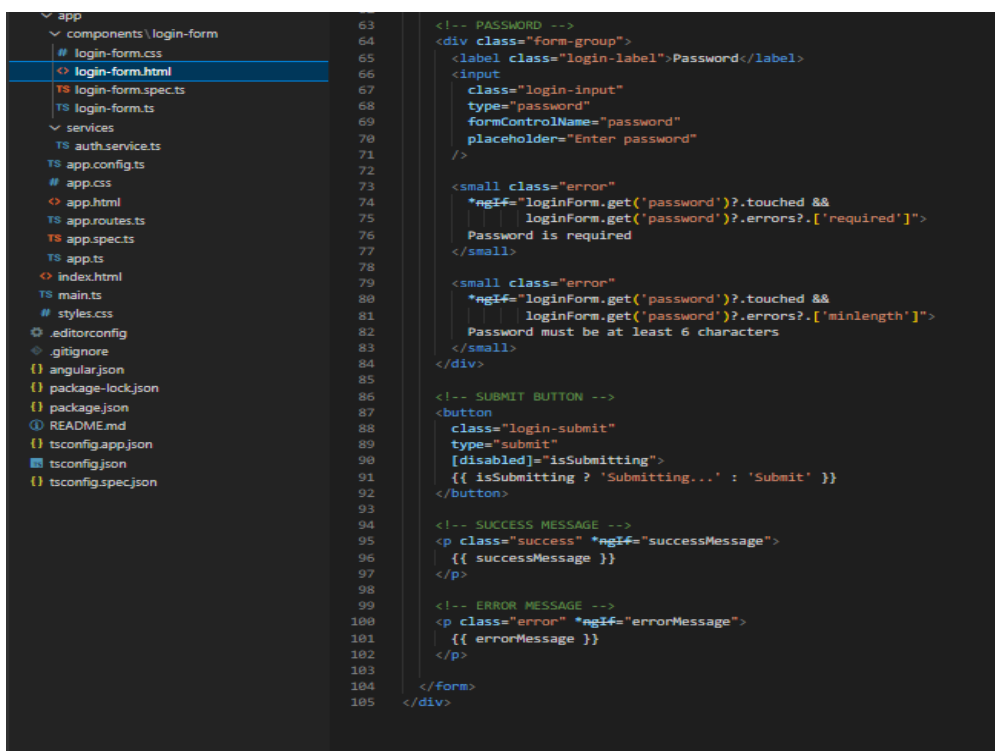
```
EXPLORER
└─ LOGINFORM-API
  └─ login
    └─ app
      └─ components\login-form
        └─ login-form.ts

login > src > app > components > login-form > TS login-form.ts > successMessage
1 import { Component, OnInit } from '@angular/core';
2 import { FormBuilder, FormGroup, ReactiveFormsModule, Validators } from '@angular/forms';
3 import { AuthService } from '../services/auth.service';
4 import { CommonModule } from '@angular/common';
5
6 @Component({
7   standalone: true,
8   selector: 'app-login',
9   templateUrl: './login-form.html',
10  styleUrls: ['./login-form.css'],
11  imports: [
12    CommonModule,
13    ReactiveFormsModule
14  ]
15 })
16 export class Login implements OnInit {
17
18   loginForm!: FormGroup;
19   isSubmitting = false;
20   successMessage = '';
21   errorMessage = '';
22
23   constructor(
24     private fb: FormBuilder,
25     private authService: AuthService
26   ) {}
27
28   ngOnInit(): void {
29     this.loginForm = this.fb.group({
30       firstName: ['', Validators.required],
31       lastName: ['', Validators.required],
32       email: ['', [Validators.required, Validators.email]],
33       password: ['', [Validators.required, Validators.minLength(6)]]
34     });
35   }
36
37   handleSubmit(): void {
38     // clear old messages
39     this.successMessage = '';
40     this.errorMessage = '';
41
42     if (this.loginForm.invalid) {
43       this.loginForm.markAllAsTouched();
44       return;
45     }
46
47     this.isSubmitting = true;
48
49     this.authService.login(this.loginForm.value).subscribe({
50       next: (res) => {
51         this.successMessage = 'Login successful!';
52         this.loginForm.reset();
53         this.isSubmitting = false;
54       },
55       error: (err) => {
56         this.errorMessage =
57           err?.error?.message || 'Login failed. Please try again.';
58         this.isSubmitting = false;
59       },
60     });
61   }
62 }
```

Step 6 : Write the HTML code for the login form in login-form.html.

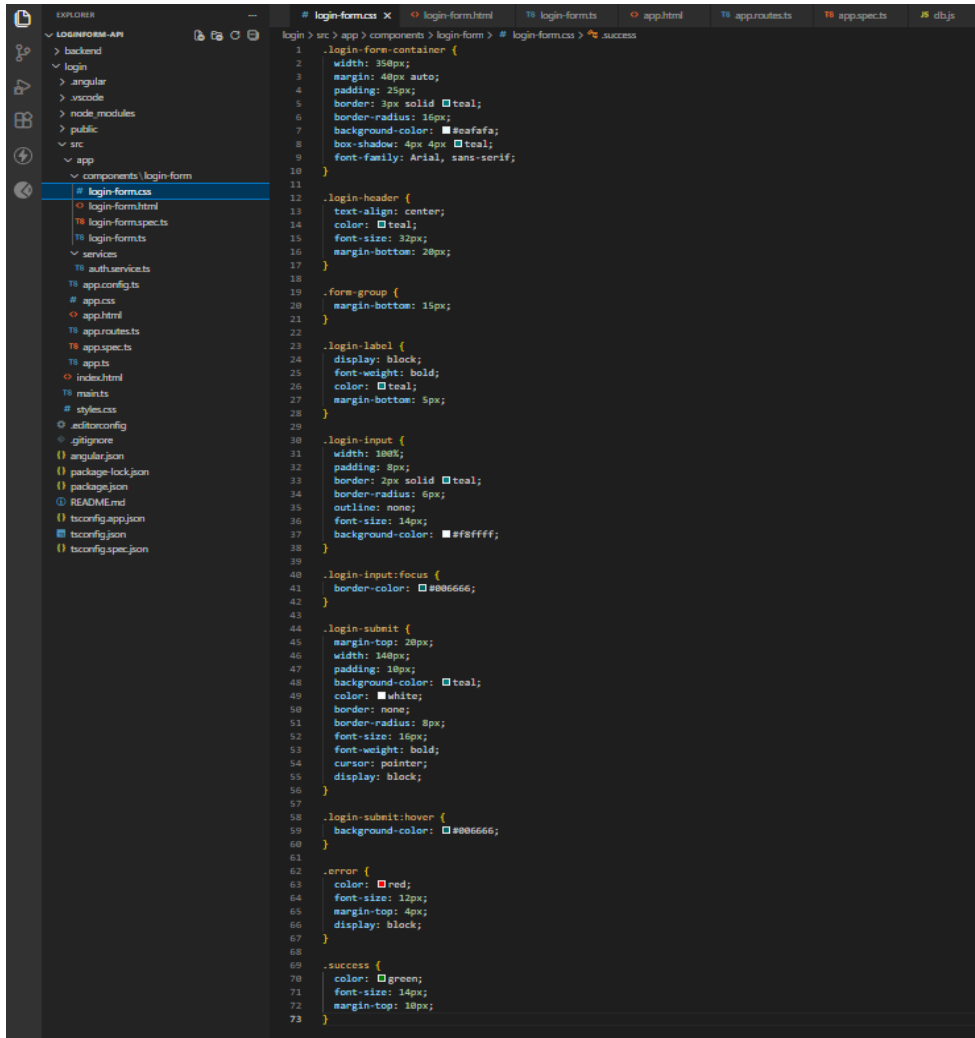


```
1  <div class="login-form-container">
2    <form [formGroup]="loginForm" (ngSubmit)="handleSubmit()">
3
4      <h1 class="login-header">LOGIN FORM</h1>
5
6      <!-- FIRST NAME -->
7      <div class="form-group">
8        <label class="login-label">First Name</label>
9        <input
10          class="login-input"
11          type="text"
12          formControlName="firstName"
13          placeholder="Enter first name"
14        />
15
16        <small class="error">
17          *ngIf="loginForm.get('firstName')?.touched &&
18            loginForm.get('firstName')?.errors?.['required']">
19            First name is required
20          </small>
21        </div>
22
23      <!-- LAST NAME -->
24      <div class="form-group">
25        <label class="login-label">Last Name</label>
26        <input
27          class="login-input"
28          type="text"
29          formControlName="lastName"
30          placeholder="Enter last name"
31        />
32
33        <small class="error">
34          *ngIf="loginForm.get('lastName')?.touched &&
35            loginForm.get('lastName')?.errors?.['required']">
36            Last name is required
37          </small>
38        </div>
39
40      <!-- EMAIL -->
41      <div class="form-group">
42        <label class="login-label">Email</label>
43        <input
44          class="login-input"
45          type="email"
46          formControlName="email"
47          placeholder="Enter email"
48        />
49
50        <small class="error">
51          *ngIf="loginForm.get('email')?.touched &&
52            loginForm.get('email')?.errors?.['required']">
53            Email is required
54          </small>
55
56        <small class="error">
57          *ngIf="loginForm.get('email')?.touched &&
58            loginForm.get('email')?.errors?.['email']">
59            Enter a valid email address
60          </small>
61        </div>
```



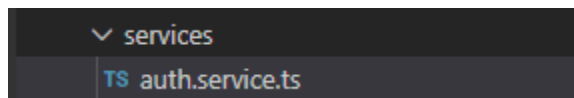
```
63
64      <!-- PASSWORD -->
65      <div class="form-group">
66        <label class="login-label">Password</label>
67        <input
68          class="login-input"
69          type="password"
70          formControlName="password"
71          placeholder="Enter password"
72        />
73
74        <small class="error">
75          *ngIf="loginForm.get('password')?.touched &&
76            loginForm.get('password')?.errors?.['required']">
77            Password is required
78          </small>
79
80        <small class="error">
81          *ngIf="loginForm.get('password')?.touched &&
82            loginForm.get('password')?.errors?.['minlength']">
83            Password must be at least 6 characters
84          </small>
85        </div>
86
87      <!-- SUBMIT BUTTON -->
88      <button
89        class="login-submit"
90        type="submit"
91        [disabled]="isSubmitting">
92        {{ isSubmitting ? 'Submitting...' : 'Submit' }}
93      </button>
94
95      <!-- SUCCESS MESSAGE -->
96      <p class="success" *ngIf="successMessage">
97        {{ successMessage }}
98      </p>
99
100     <!-- ERROR MESSAGE -->
101     <p class="error" *ngIf="errorMessage">
102       {{ errorMessage }}
103     </p>
104   </form>
105 </div>
```

Step 7 : Write the CSS code for styling the login form in login-form.css.

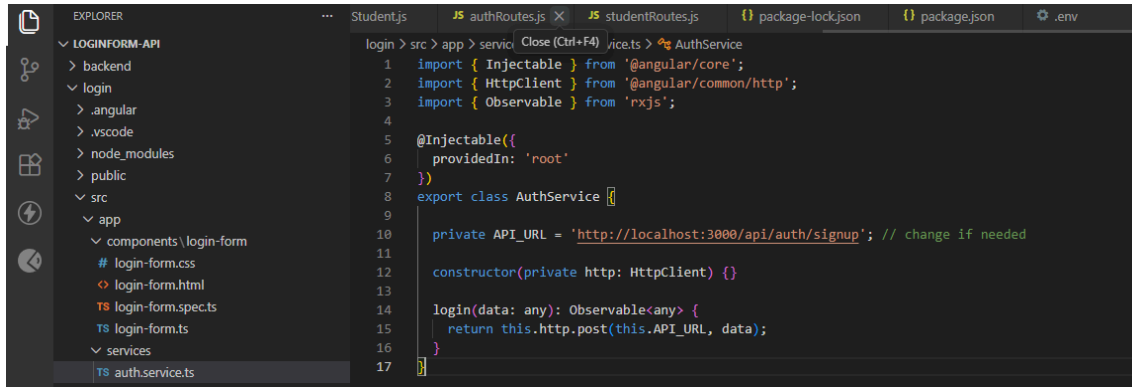


```
1 .login-form-container {
2   width: 350px;
3   margin: 40px auto;
4   padding: 25px;
5   border: 2px solid #008080;
6   border-radius: 10px;
7   background-color: #f0f0f0;
8   box-shadow: 4px 4px #008080;
9   font-family: Arial, sans-serif;
10 }
11
12 .login-header {
13   text-align: center;
14   color: #008080;
15   font-size: 32px;
16   margin-bottom: 20px;
17 }
18
19 .form-group {
20   margin-bottom: 15px;
21 }
22
23 .login-label {
24   display: block;
25   font-weight: bold;
26   color: #008080;
27   margin-bottom: 5px;
28 }
29
30 .login-input {
31   width: 280px;
32   padding: 8px;
33   border: 2px solid #008080;
34   border-radius: 6px;
35   outline: none;
36   font-size: 14px;
37   background-color: #ffffff;
38 }
39
40 .login-input:focus {
41   border-color: #008080;
42 }
43
44 .login-submit {
45   margin-top: 20px;
46   width: 140px;
47   padding: 10px;
48   background-color: #008080;
49   color: white;
50   border: none;
51   border-radius: 8px;
52   font-size: 16px;
53   font-weight: bold;
54   cursor: pointer;
55   display: block;
56 }
57
58 .login-submit:hover {
59   background-color: #006666;
60 }
61
62 .error {
63   color: red;
64   font-size: 12px;
65   margin-top: 4px;
66   display: block;
67 }
68
69 .success {
70   color: green;
71   font-size: 14px;
72   margin-top: 10px;
73 }
```

Step 8: Create a services folder in the project

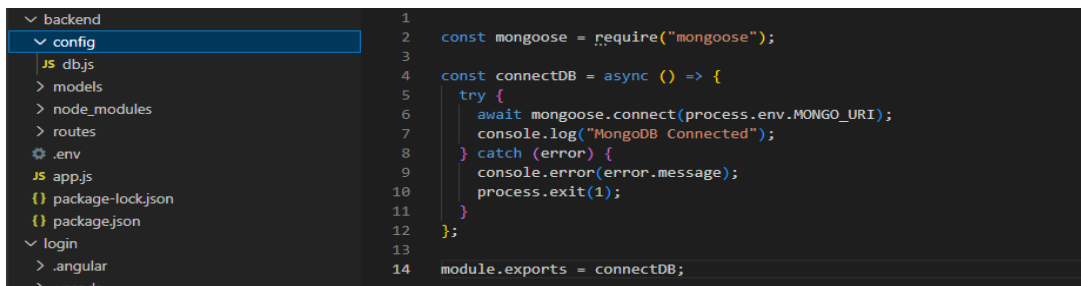
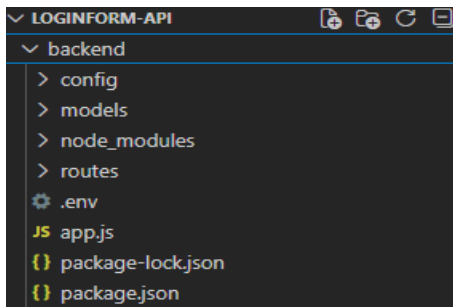


Step 9: Create and write code in auth.service.ts

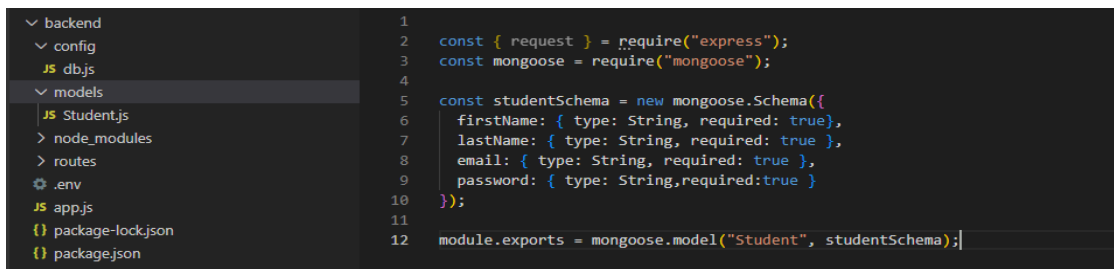


```
login > src > app > service Close (Ctrl+F4) /ice.ts > AuthService
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class AuthService {
9
10   private API_URL = 'http://localhost:3000/api/auth/signup'; // change if needed
11
12   constructor(private http: HttpClient) {}
13
14   login(data: any): Observable<any> {
15     return this.http.post(this.API_URL, data);
16   }
17 }
```

Step 10: Create a separate folder for the backend.



```
1
2 const mongoose = require("mongoose");
3
4 const connectDB = async () => {
5   try {
6     await mongoose.connect(process.env.MONGO_URI);
7     console.log("MongoDB Connected");
8   } catch (error) {
9     console.error(error.message);
10    process.exit(1);
11  }
12 };
13
14 module.exports = connectDB;
```



```
1
2 const { request } = require("express");
3 const mongoose = require("mongoose");
4
5 const studentSchema = new mongoose.Schema({
6   firstName: { type: String, required: true },
7   lastName: { type: String, required: true },
8   email: { type: String, required: true },
9   password: { type: String, required: true }
10 });
11
12 module.exports = mongoose.model("Student", studentSchema);
```

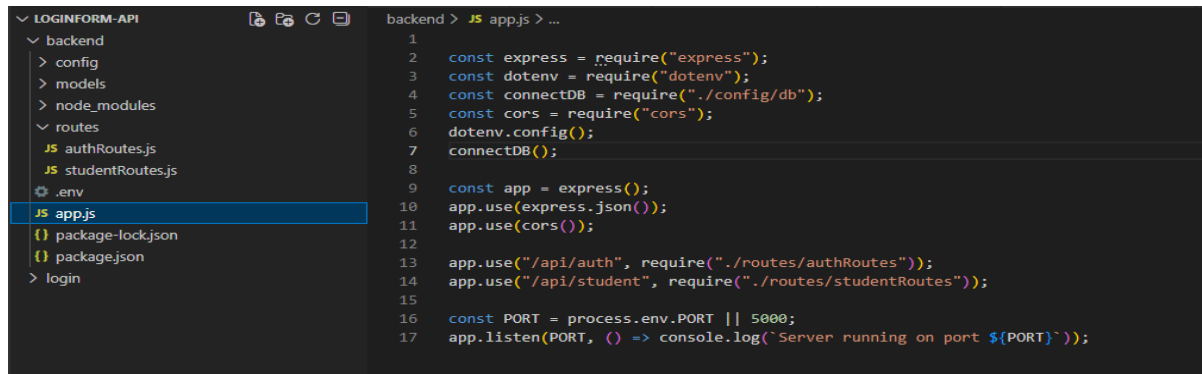
```
1
2 const express = require("express");
3 const Student = require("../models/student");
4 const router = express.Router();
5
6 router.post("/signup", async (req, res) => {
7   try {
8     const student = await Student.create(req.body);
9     res.status(201).json(student);
10  } catch (err) {
11    res.status(400).json({ error: err.message });
12  }
13 });
14
15 router.post("/login", async (req, res) => {
16   try {
17     const { email, password } = req.body;
18     const student = await Student.findOne({
19       email: email,
20       password: password
21     }).select({
22       password: 0
23     });
24
25     if (!student) {
26       res.status(400).json({
27         message: "invalid email or password"
28       });
29     }
30
31     res.status(201).json({
32       data: student
33     });
34   } catch (err) {
35     res.status(400).json({ error: err.message });
36   }
37 });
38
39
40
41 module.exports = router;
```

```
1
2 const express = require("express");
3 const Student = require("../models/student");
4 const router = express.Router();
5
6
7
8 router.get("/", async (req, res) => {
9   const students = await Student.find()
10     .select({
11       password: 0
12     });
13   res.json(
14     {
15       data: students
16     }
17   );
18 });
19
20 module.exports = router;
```

Step 11: Create a .env file to store environment variables.

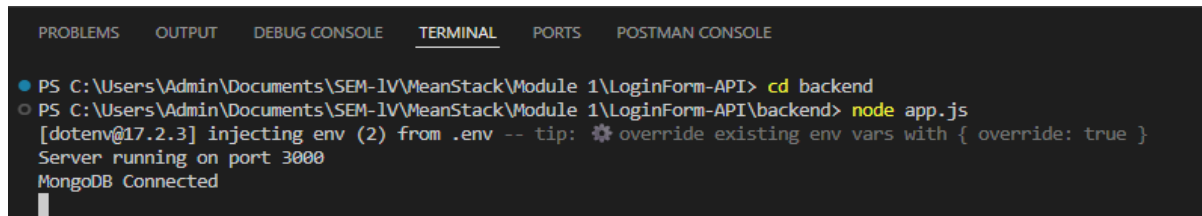
```
backend > .env
1 PORT=3000
2 MONGO_URI=mongodb://127.0.0.1:27017/collegeDB
3
```

Step 12: Create an app.js file for backend server logic.



```
backend > JS app.js > ...
1
2   const express = require("express");
3   const dotenv = require("dotenv");
4   const connectDB = require("../config/db");
5   const cors = require("cors");
6   dotenv.config();
7   connectDB();
8
9   const app = express();
10  app.use(express.json());
11  app.use(cors());
12
13  app.use("/api/auth", require("../routes/authRoutes"));
14  app.use("/api/student", require("../routes/studentRoutes"));
15
16  const PORT = process.env.PORT || 5000;
17  app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Step 13: Run the backend server

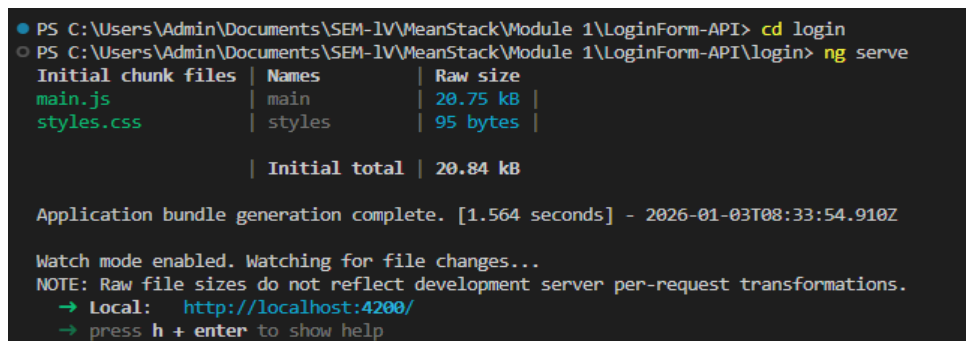


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE

● PS C:\Users\Admin\Documents\SEM-1V\MeanStack\Module 1\LoginForm-API> cd backend
○ PS C:\Users\Admin\Documents\SEM-1V\MeanStack\Module 1\LoginForm-API\backend> node app.js
[dotenv@17.2.3] injecting env (2) from .env -- tip: ⚙ override existing env vars with { override: true }
Server running on port 3000
MongoDB Connected
```

Step 14: Verify that the login form is working correctly.

Step 15: Run the frontend application.



```
● PS C:\Users\Admin\Documents\SEM-1V\MeanStack\Module 1\LoginForm-API> cd login
○ PS C:\Users\Admin\Documents\SEM-1V\MeanStack\Module 1\LoginForm-API\login> ng serve

Initial chunk files | Names | Raw size
main.js             | main  | 20.75 kB
styles.css          | styles | 95 bytes
| Initial total | 20.84 kB

Application bundle generation complete. [1.564 seconds] - 2026-01-03T08:33:54.910Z

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
→ Local: http://localhost:4200/
→ press h + enter to show help
```

LOGIN FORM

First Name

Last Name

Email

Password

Submit

Login successful!

Step 16: Check the database to confirm whether the user credentials are stored

