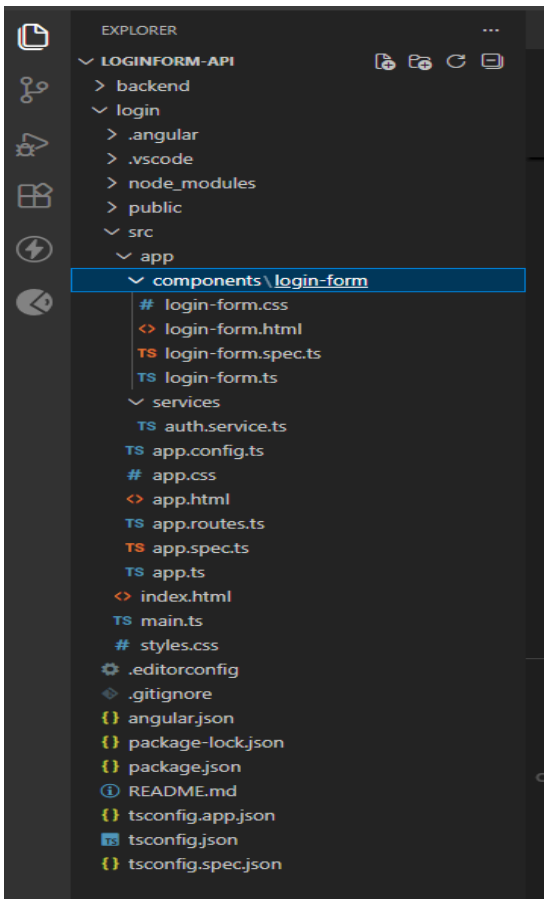
Practical No : 05

Aim : - Connecting Angular Login Form to rest API

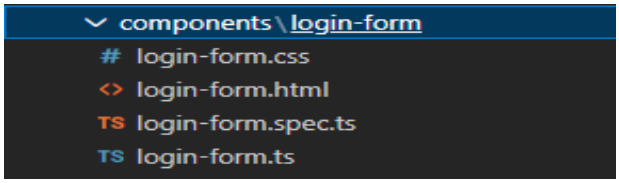
Step 1 : Creating a new file



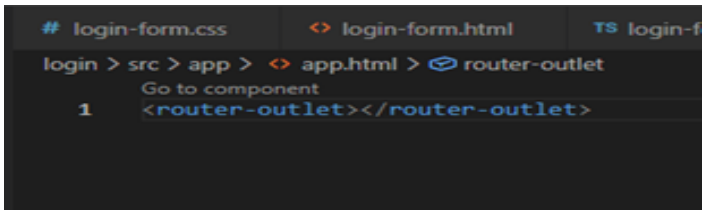
Step 2 : Open the terminal and run the required commands

```
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)
```

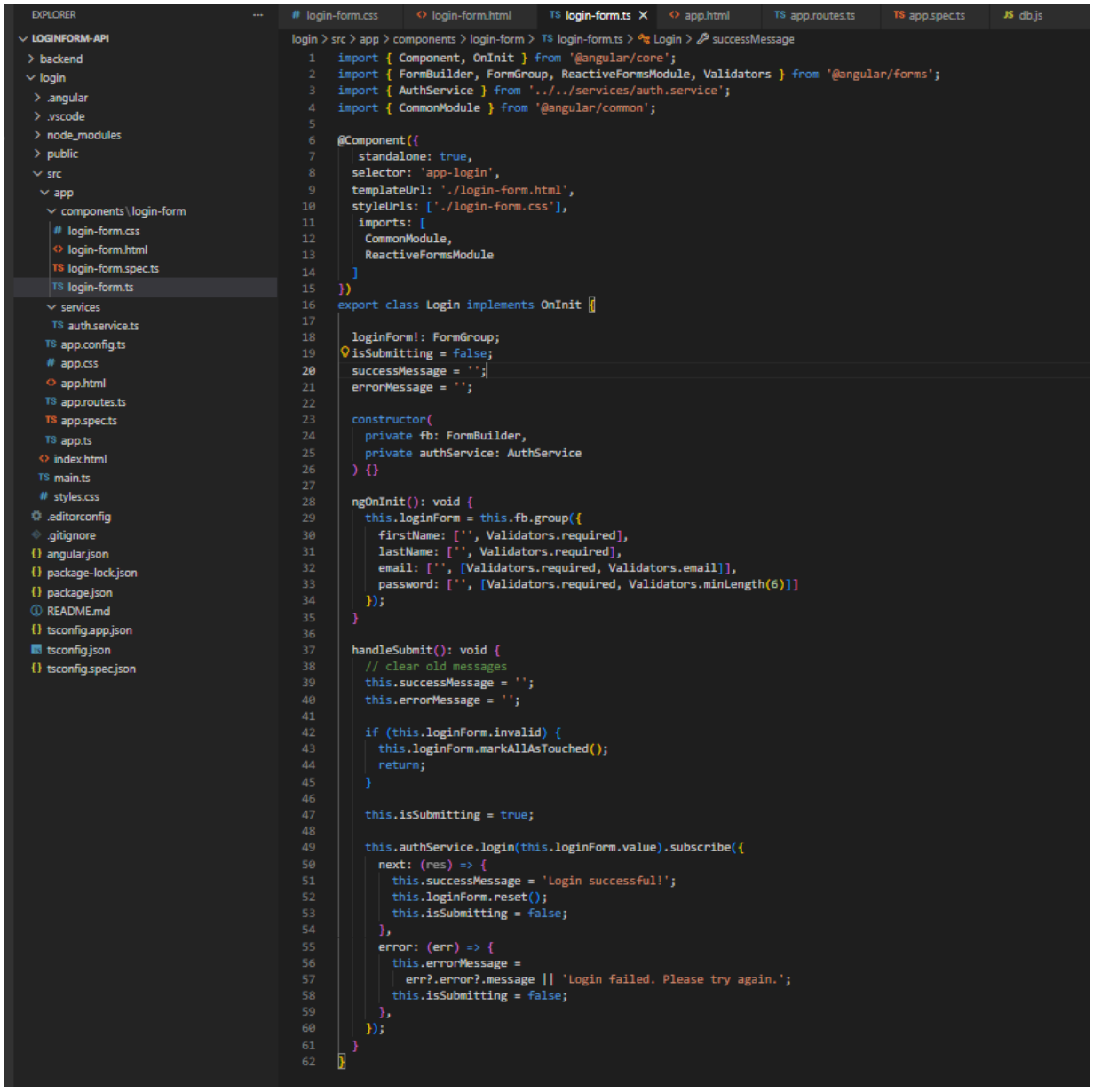
Step 3 : Create the necessary components



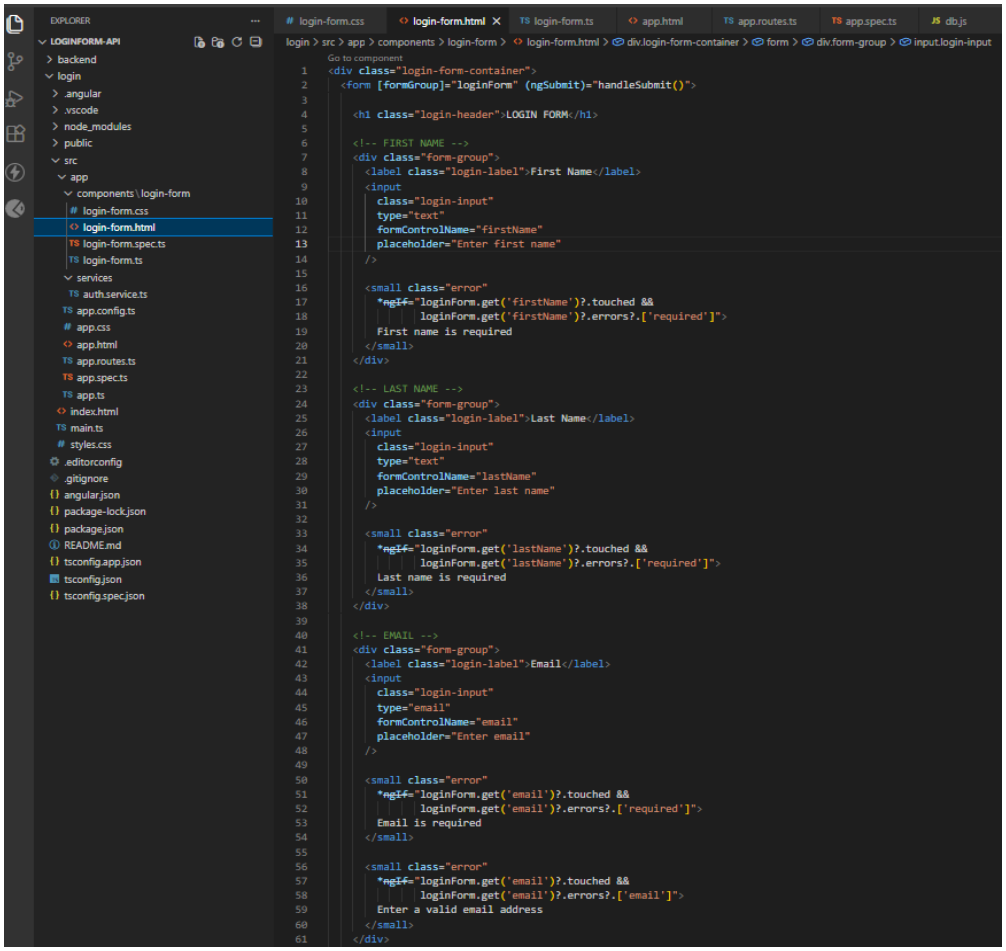
Step 4: Edit the app.html file to define the main structure



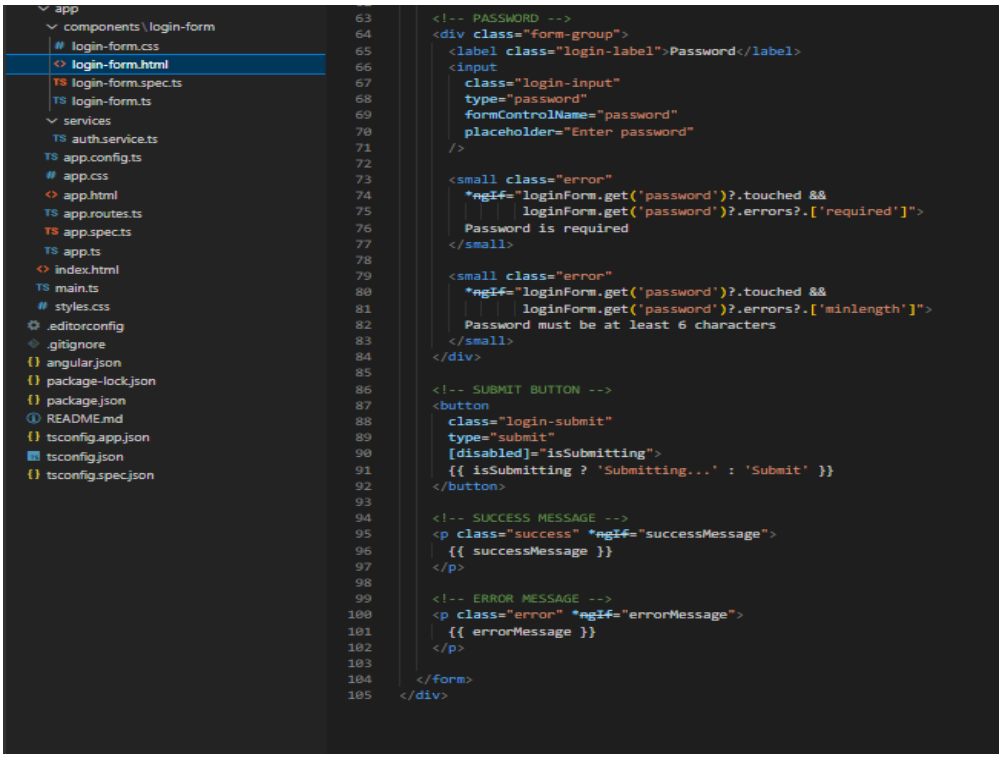
Step 5 : Edit the `login-form.ts` file to write the logic



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

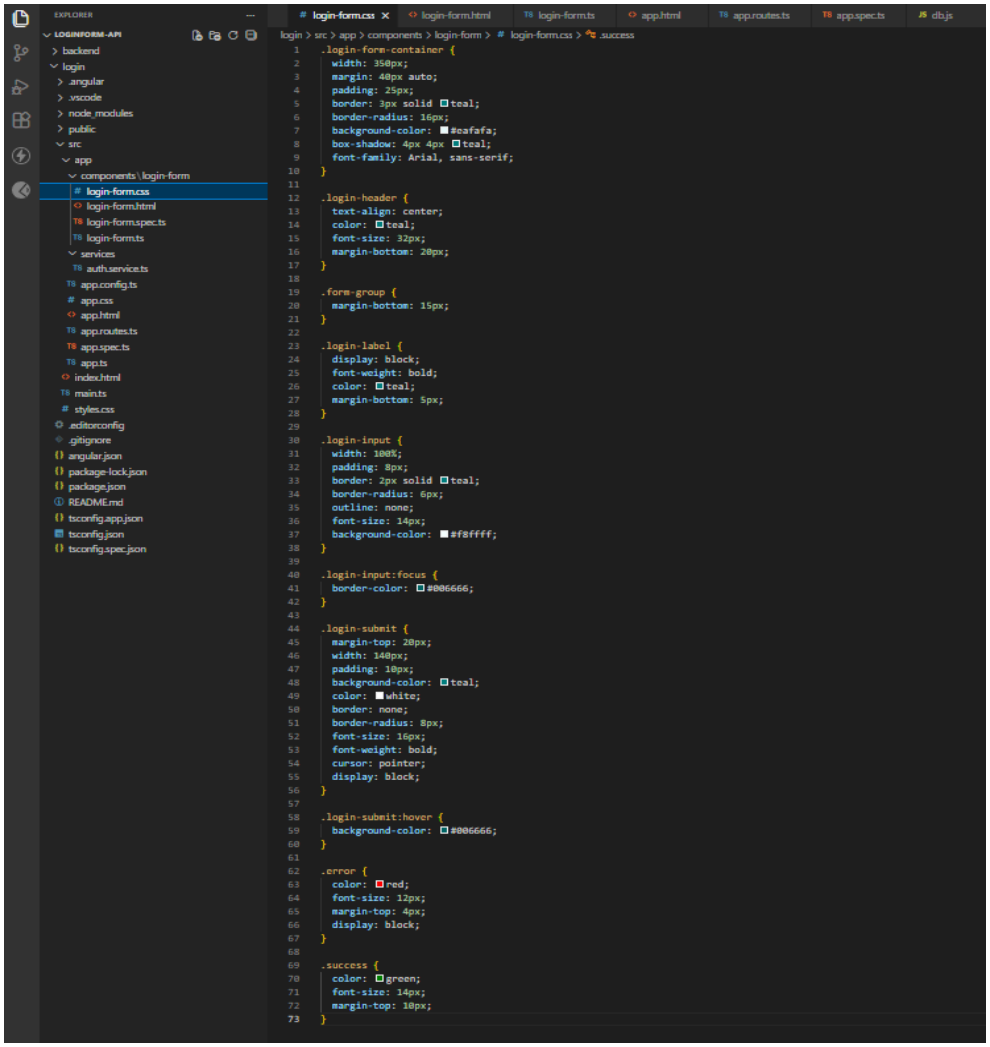


```
<div class="login-form-container">
  <form [formGroup]="loginForm" (ngSubmit)="handleSubmit()">

    <h1 class="login-header">LOGIN FORM</h1>

    <!-- FIRST NAME -->
    <div class="form-group">
      <label class="login-label">First Name</label>
      <input
        class="login-input"
        type="text"
        formControlName="firstName"
        placeholder="Enter first name"
      />

      <small class="error"
        *ngIf="loginForm.get('firstName')?.touched &&
               loginForm.get('firstName')?.errors?.['required']">
        First name is required
      </small>
    </div>

    <!-- LAST NAME -->
    <div class="form-group">
      <label class="login-label">Last Name</label>
      <input
        class="login-input"
        type="text"
        formControlName="lastName"
        placeholder="Enter last name"
      />

      <small class="error"
        *ngIf="loginForm.get('lastName')?.touched &&
               loginForm.get('lastName')?.errors?.['required']">
        Last name is required
      </small>
    </div>

    <!-- EMAIL -->
    <div class="form-group">
      <label class="login-label">Email</label>
      <input
        class="login-input"
        type="email"
        formControlName="email"
        placeholder="Enter email"
      />

      <small class="error"
        *ngIf="loginForm.get('email')?.touched &&
               loginForm.get('email')?.errors?.['required']">
        Email is required
      </small>

      <small class="error"
        *ngIf="loginForm.get('email')?.touched &&
               loginForm.get('email')?.errors?.['email']">
        Enter a valid email address
      </small>
    </div>
```



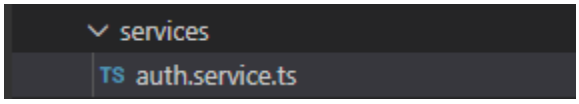
```
    <!-- PASSWORD -->
    <div class="form-group">
      <label class="login-label">Password</label>
      <input
        class="login-input"
        type="password"
        formControlName="password"
        placeholder="Enter password"
      />

      <small class="error"
        *ngIf="loginForm.get('password')?.touched &&
               loginForm.get('password')?.errors?.['required']">
        Password is required
      </small>

      <small class="error"
        *ngIf="loginForm.get('password')?.touched &&
               loginForm.get('password')?.errors?.['minlength']">
        Password must be at least 6 characters
      </small>
    </div>

    <!-- SUBMIT BUTTON -->
    <button
      class="login-submit"
      type="submit"
      [disabled]="isSubmitting">
      {{ isSubmitting ? 'Submitting...' : 'Submit' }}
    </button>

    <!-- SUCCESS MESSAGE -->
    <p class="success" *ngIf="successMessage">
      {{ successMessage }}
    </p>

    <!-- ERROR MESSAGE -->
    <p class="error" *ngIf="errorMessage">
      {{ errorMessage }}
    </p>

  </form>
</div>
```

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

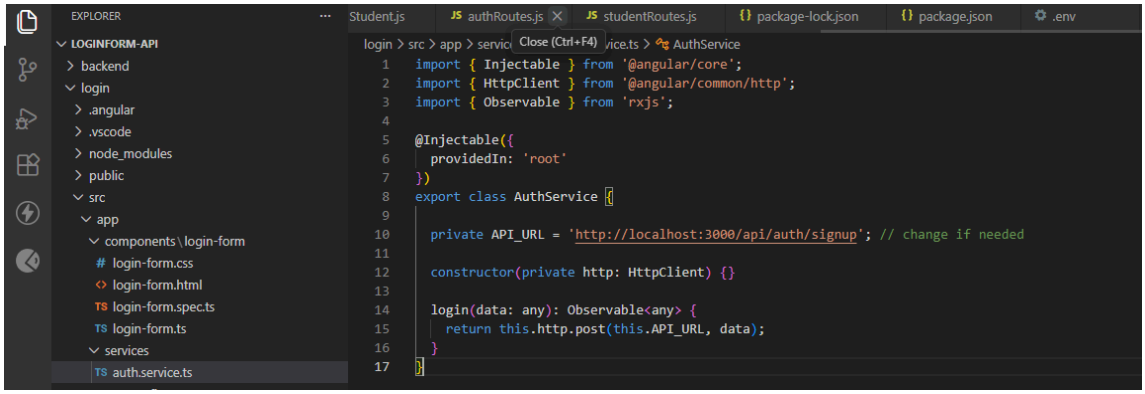


```css
.login-form-container {
  width: 350px;
  margin: 40px auto;
  padding: 25px;
  border: 3px solid teal;
  border-radius: 16px;
  background-color: #eafafa;
  box-shadow: 4px 4px teal;
  font-family: Arial, sans-serif;
}

.login-header {
  text-align: center;
  color: teal;
  font-size: 32px;
  margin-bottom: 20px;
}

.form-group {
  margin-bottom: 15px;
}

.login-label {
  display: block;
  font-weight: bold;
  color: teal;
  margin-bottom: 5px;
}

.login-input {
  width: 100%;
  padding: 8px;
  border: 2px solid teal;
  border-radius: 6px;
  outline: none;
  font-size: 14px;
  background-color: #f8ffff;
}

.login-input:focus {
  border-color: #006666;
}

.login-submit {
  margin-top: 20px;
  width: 140px;
  padding: 10px;
  background-color: teal;
  color: white;
  border: none;
  border-radius: 8px;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  display: block;
}

.login-submit:hover {
  background-color: #006666;
}

.error {
  color: red;
  font-size: 12px;
  margin-top: 4px;
  display: block;
}

.success {
  color: green;
  font-size: 14px;
  margin-top: 10px;
}
```

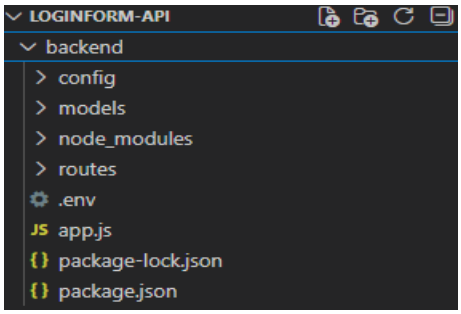
Step 8: Create a services folder in the project

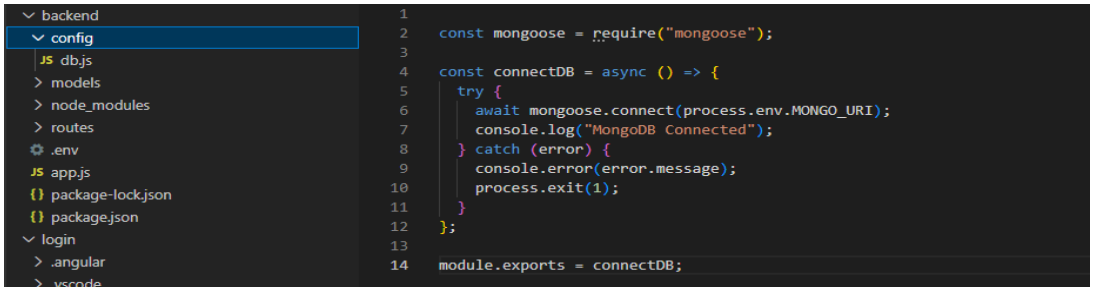


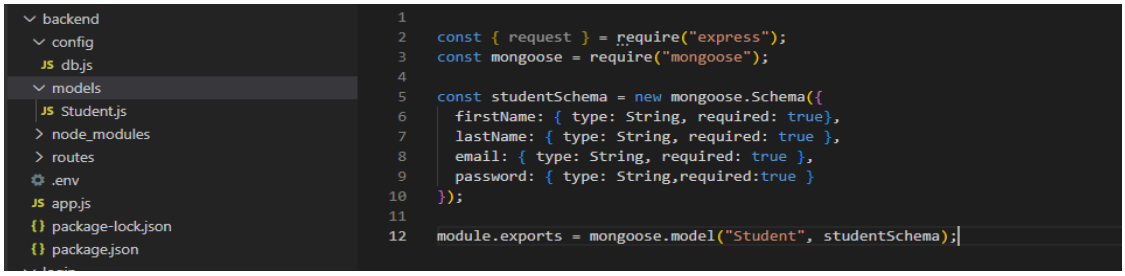
Step 9: Create and write code in auth.service.ts

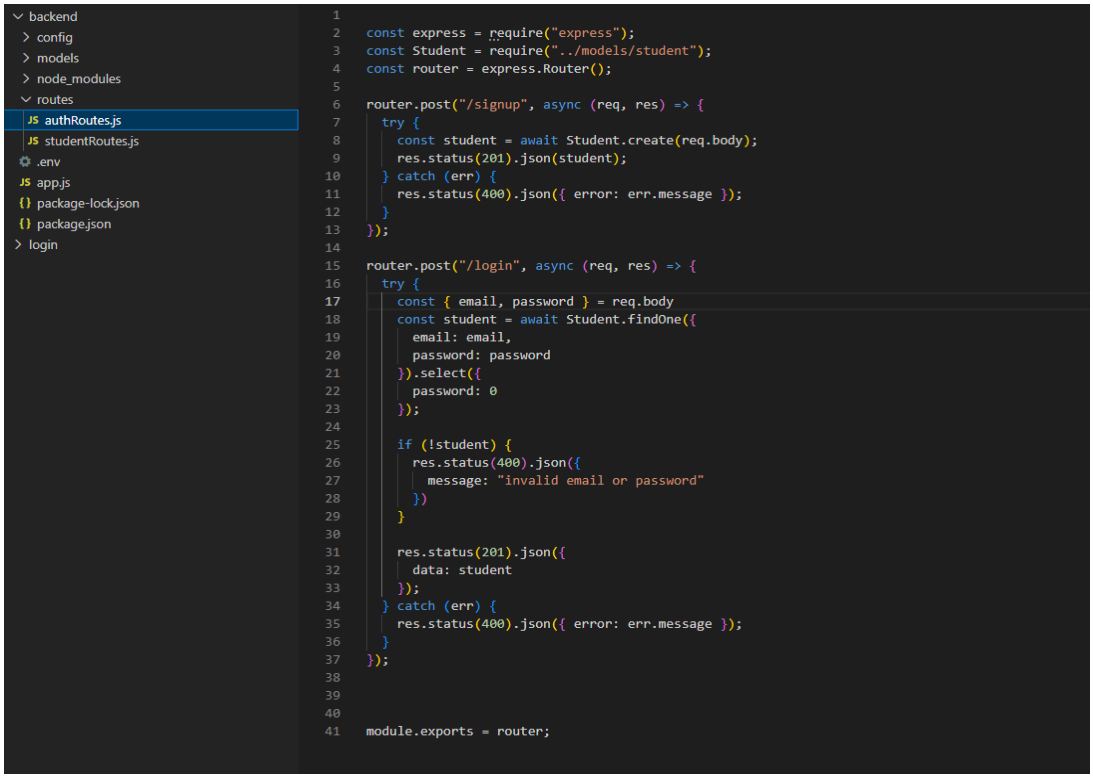


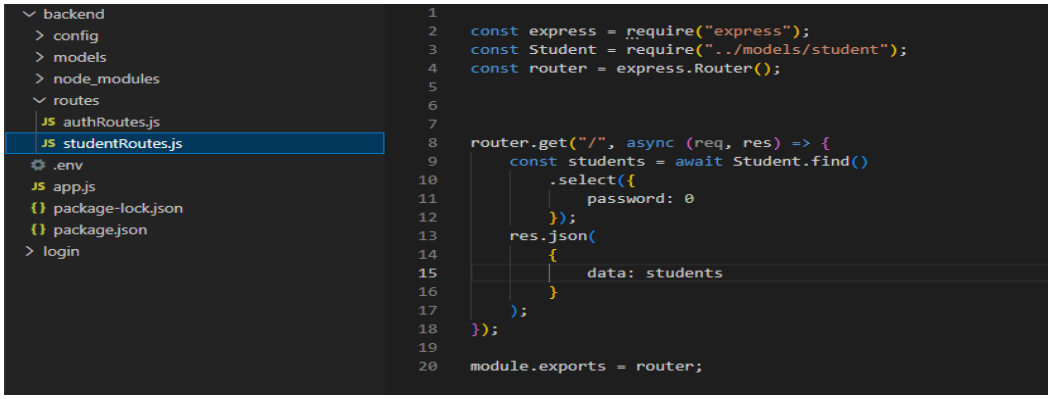
Step 10: Create a separate folder for the backend.



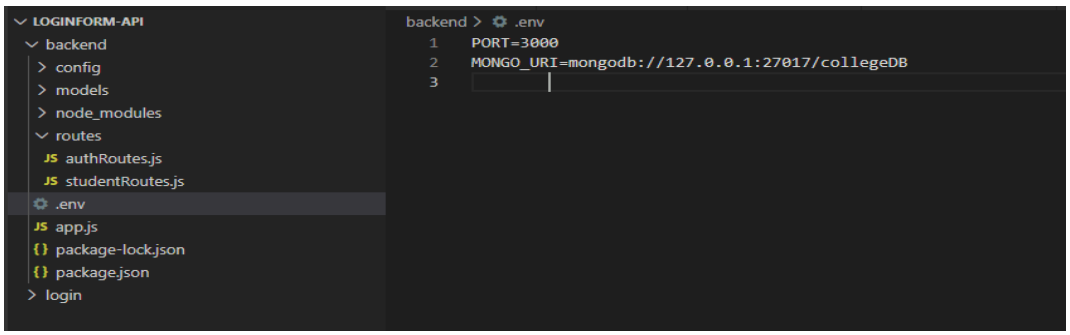




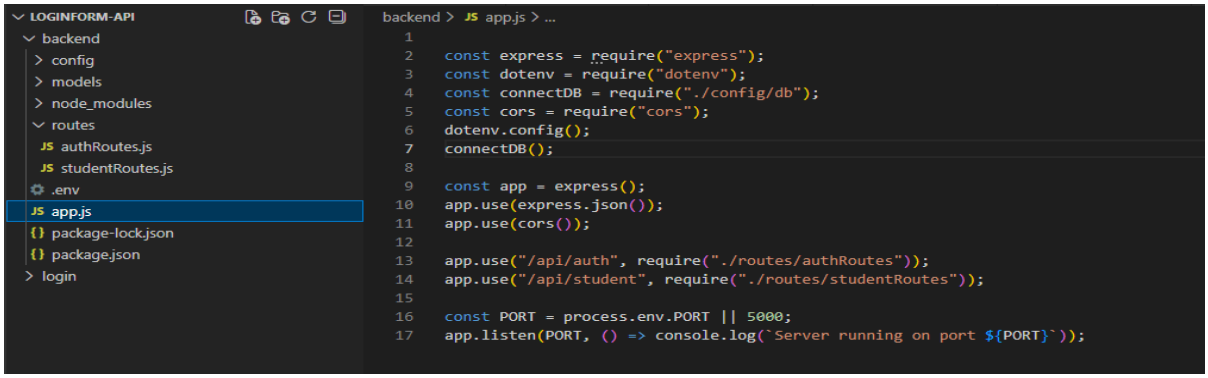




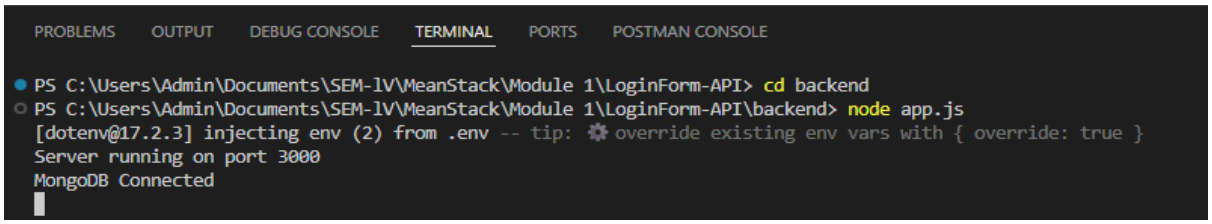
Step 11: Create a .env file to store environment variables.



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

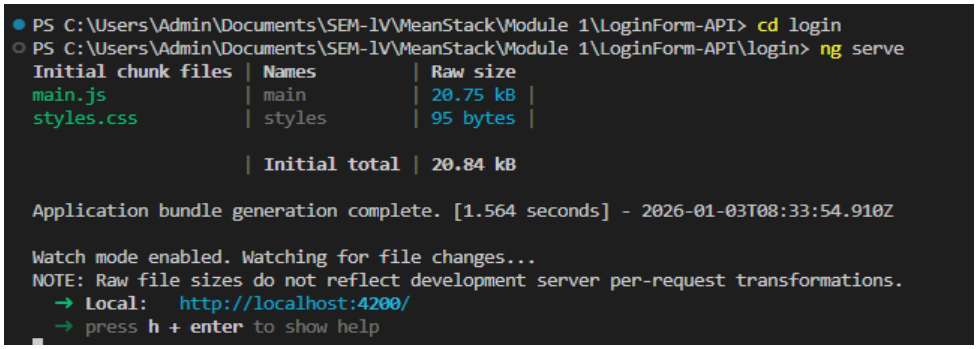


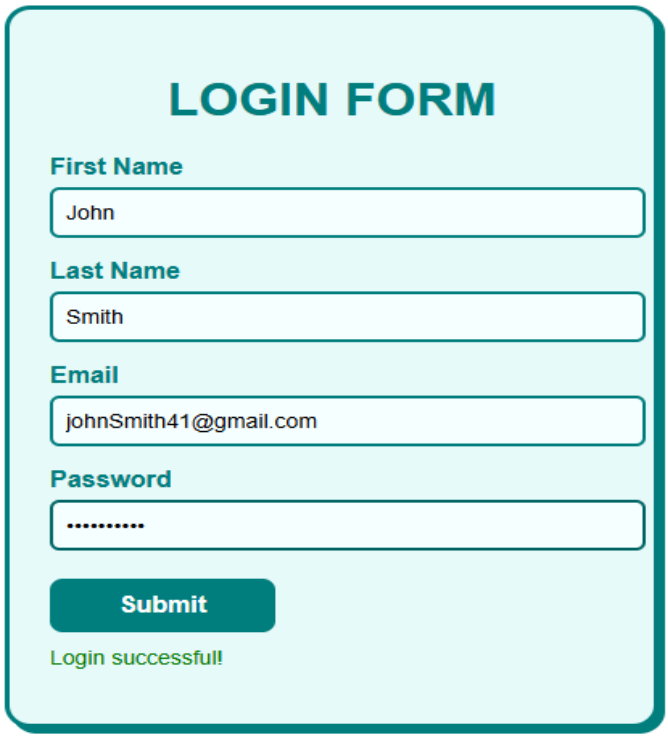
Step 13: Run the backend server



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

Step 15: Run the frontend application.





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