SOURCE CODE

```
package com.example.Medicare;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class MedicareApplication {
public static void main(String[] args) {
SpringApplication.run(MedicareApplication.class, args);
}
}
```

```
package com.example.Medicare.controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
```

```
import org.springframework.web.bind.annotation.RestController;

import com.example.Medicare.entity.Admin;
import com.example.Medicare.exception.DuplicateEmailException;
import com.example.Medicare.service.AdminService;



@RestController
@CrossOrigin(origins = "http://localhost:4200")
public class AdminController {
	@Autowired
	AdminService adminService;

	@PostMapping("adminLogin")
	public Admin adminCredentials(@RequestBody Admin ad,
HttpServletRequest request) throws DuplicateEmailException {

		String name = "Sporty Shoes";
		String email = "simplilearn@gmail.com";
		String password = "admin";
		Admin admin = new Admin();
		admin.setName(name);
		admin.setEmail(email);
```

```
            admin.setPassword(password);
            try {
                adminService.create(admin);
            } catch (DuplicateEmailException e) {
                e.getMessage();
            }

            Admin adminget =
adminService.find("simplilearn@gmail.com");
            System.out.println(adminget);

            if ((adminget.getEmail()).equals(ad.getEmail()) &&
(adminget.getPassword()).equals(ad.getPassword())) {
                HttpSession usersession = request.getSession();
                usersession.setAttribute("LoginCredentials", adminget);
                return adminget;
            } else {
                return null;
            }

        }

        @PutMapping("/changePassword")
```

```
	public Admin changePassword(@RequestParam("password")
String password,
			@RequestParam("newPassword") String newPassword,
HttpServletRequest request) {
		HttpSession session = request.getSession();


		Admin admin = (Admin)
session.getAttribute("LoginCredentials");
		try {
			admin = adminService.find(admin.getEmail());
		} catch (DuplicateEmailException e1) {
			e1.printStackTrace();
		}


		try {
			adminService.changePassword(1, password,
newPassword);


		} catch (Exception e) {
			e.printStackTrace();
		}


		return admin;
	}
```

```
}

package com.example.Medicare.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.example.Medicare.entity.Cart;
import com.example.Medicare.entity.Medicine;
import com.example.Medicare.entity.User;
import com.example.Medicare.service.CartService;
```

```
import com.example.Medicare.service.MedicineService;
import com.example.Medicare.service.UserService;

@RestController
@CrossOrigin(origins = "http://localhost:4200")
public class CartController {

	@Autowired
	CartService cartService;

	@Autowired
	MedicineService medicineService;



	@PostMapping("/addCart/{id}")
	public Cart addItem(@PathVariable int id,HttpServletRequest request) {
		HttpSession session= request.getSession();
		User user = (User) session.getAttribute("LoginCredentials");

		Medicine medicine = medicineService.getMedicine(id);

		Cart cart = cartService.add(user, medicine);
```

```
        return cart;

    }



    @GetMapping("/getCart")
    public List<Cart> getCartItems(HttpServletRequest request){
        HttpSession session= request.getSession();
        User user = (User) session.getAttribute("LoginCredentials");

        List<Cart> cartItems = cartService.getCartItems(user);

        return cartItems;
    }



    @DeleteMapping("/deleteCart/{id}")
    public String deleteCartItem(@PathVariable int id) {
        String str = cartService.deleteCartItem(id);
        return str;
    }
```

```
}
package com.example.Medicare.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.example.Medicare.entity.Medicine;
import com.example.Medicare.service.MedicineService;

@RestController
@CrossOrigin(origins = "http://localhost:4200")
public class MedicineController {
	@Autowired
	MedicineService medicineService;
```

```
    @PostMapping("/addMedicine")
    public Medicine createMedicine(@RequestBody Medicine med) {
        Medicine medicine = medicineService.create(med);
        return medicine;
    }

    @PostMapping("/updateMedicine/{id}")
    public Medicine updateMedicine(@RequestBody Medicine med,
@PathVariable int id) {
        System.out.println("Id for the update" + id);
        Medicine medicine = medicineService.update(id, med);
        return medicine;
    }

    @DeleteMapping("/deleteMedicine/{id}")
    public String deleteMedicine(@PathVariable int id) {
        String message = medicineService.delete(id);
        return message;
    }

    @GetMapping("/getMedicine")
    public List<Medicine> getMedicine() {
        List<Medicine> medicines =
medicineService.getMedicines();
```

```
            return medicines;
        }
}
package com.example.Medicare.controller;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.example.Medicare.entity.User;
import com.example.Medicare.exception.DuplicateEmailException;
import com.example.Medicare.service.UserService;
@RestController
@CrossOrigin(origins = "http://localhost:4200")
public class UserController {
@Autowired
UserService userService;
@PostMapping("/signup")
```

```
public User createUser(@RequestBody User user) {
User newUser = null;
try {
newUser = userService.create(user);
} catch (DuplicateEmailException e) {
e.printStackTrace();
}
return newUser;
}
@PostMapping("/login")
public User loginUser(@RequestParam("email") String email,
@RequestParam("password") String password) {
User user = userService.login(email, password);
return user;
}
// @PostMapping("/login")
// public ResponseEntity<String> login(@RequestParam String email,
@RequestParam String password, HttpServletRequest request) {
// // Assuming you have a userService that performs user authentication
// User user = userService.authenticateUser(email, password);
//
// if (user != null) {
// // Set the user's login credentials in the session
// HttpSession session = request.getSession();
```

```
// session.setAttribute("LoginCredentials", user);
//
// return ResponseEntity.ok("Login successful");
// } else {
// return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid username or password");
// }
// }
}


package com.example.Medicare.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="medicine")

public class Medicine {
	@Id
```

```
@GeneratedValue(strategy=GenerationType.IDENTITY)
private int id;

private String name;

private String price;

private String seller;

private String productDescription;

private int offers;

private boolean isEnabled;

public int getId() {
	return id;
}

public void setId(int id) {
	this.id = id;
}
```

```
public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getPrice() {
        return price;
}

public void setPrice(String price) {
        this.price = price;
}

public String getSeller() {
        return seller;
}

public void setSeller(String seller) {
        this.seller = seller;
}
```

```
public String getProductDescription() {
        return productDescription;
}

public void setProductDescription(String productDescription) {
        this.productDescription = productDescription;
}

public int getOffers() {
        return offers;
}

public void setOffers(int offers) {
        this.offers = offers;
}

public boolean isEnabled() {
        return isEnabled;
}

public void setEnabled(boolean isEnabled) {
        this.isEnabled = isEnabled;
```

```
	}

	@Override
	public String toString() {
		return "Medicine [id=" + id + ", name=" + name + ", price="
+ price + ", seller=" + seller
				+ ", productDescription=" + productDescription +
", offers=" + offers + ", isEnabled=" + isEnabled
				+ "]";
	}

}
package com.example.Medicare.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class User {
	@Override
	public String toString() {
```

```
        return "User [id=" + id + ", name=" + name + ", email=" +
email + ", phone=" + phone + ", password=" + password
                + ", address=" + address + "]";
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "User_Name")
    private String name;

    @Column(name = "Email")
    private String email;

    @Column(name = "Phone")
    private String phone;

    @Column(name = "Password")
    private String password;

    @Column(name = "Address")
    private String address;
```

```
public int getId() {
        return id;
}

public void setId(int id) {
        this.id = id;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public String getEmail() {
        return email;
}

public void setEmail(String email) {
        this.email = email;
```

```
}

public String getPhone() {
        return phone;
}

public void setPhone(String phone) {
        this.phone = phone;
}

public String getPassword() {
        return password;
}

public void setPassword(String password) {
        this.password = password;
}

public String getAddress() {
        return address;
}

public void setAddress(String address) {
```

```
        this.address = address;
    }

}
```

```
<div class="admin-login">
  <h2>Admin Login</h2>
  <form (ngSubmit)="adminLogin()">
    <label>Email:</label>
    <input type="email" [(ngModel)]="admin.email" name="email" required />
    <label>Password:</label>
    <input
      type="password"
      [(ngModel)]="admin.password"
      name="password"
      required
    />
    <button type="submit">Login</button>
    <p class="error-message" *ngIf="loginError">{{ loginError }}</p>
  </form>
</div>
```

```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';

@Component({
  selector: 'app-admin',
  templateUrl: './admin.component.html',
  styleUrls: ['./admin.component.css'],
})
export class AdminComponent implements OnInit {
  admin = { email: '', password: '' };
  loginError: string = '';

  constructor(private http: HttpClient, private router: Router) {}
  ngOnInit(): void {
    throw new Error('Method not implemented.');
  }
```

```
  adminLogin() {
    const url = 'http://localhost:8080/adminLogin';

    this.http.post<any>(url, this.admin).subscribe(
      (response) => {
        if (response) {
          // Handle successful login
          console.log('Login successful');
          console.log('Response:', response);

          // Store the login credentials in local storage or session storage
          localStorage.setItem('adminCredentials', JSON.stringify(response));

          // Redirect to the admin dashboard or perform other actions
          this.router.navigate(['/dashboard']);
          // ...
        } else {
          // Handle login failure
          this.loginError = 'Incorrect credentials. Please try again.';
        }
      },
      (error) => {
        // Handle login error
        console.error('An error occurred during login:', error);
      }
    );
  }
}
```

```
<div class="cart-container" *ngIf="cartItems">
  <h2>Cart Details:</h2>
  <ul>
    <li *ngFor="let item of cartItems">
      <p>Medicine Name: {{ item.medicine.name }}</p>
      <p>Price: ${{ item.medicine.price }}</p>
      <p>Seller: {{ item.medicine.seller }}</p>
      <button (click)="deleteCartItem(item.id)">Delete</button>
    </li>
  </ul>
  <p class="total-price">Total Price: ${{ calculateTotalPrice() }}</p>
</div>
```

```
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';

interface CartItem {
  id: number;
  user: any;
  medicine: {
    id: number;
    name: string;
    price: string;
    seller: string;
    productDescription: any;
    offers: number;
    enabled: boolean;
  };
}

@Component({
  selector: 'app-cart',
  templateUrl: './cart.component.html',
  styleUrls: ['./cart.component.css'],
})
export class CartComponent implements OnInit {
  cartItems: any[] = [];

  constructor(private http: HttpClient) {}
  ngOnInit() {
    this.fetchCart();
  }

  fetchCart() {
    const url = 'http://localhost:8080/getCart';

    this.http.get<any[]>(url).subscribe(
      (response) => {
        this.cartItems = response;
      },
      (error) => {
        console.log('Error fetching cart details:', error);
      }
    );
  }
  calculateTotalPrice(): number {
    let totalPrice = 0;
    if (this.cartItems) {
```

```
      for (const item of this.cartItems) {
        if (item && item.medicine && item.medicine.price) {
          const formattedPrice = parseFloat(
            item.medicine.price.replace(/[^0-9.-]+/g, '')
          );
          totalPrice += formattedPrice;
        }
      }
    }
    return totalPrice;
  }

  deleteCartItem(itemId: number) {
    this.http
      .delete(`http://localhost:8080/deleteCart/${itemId}`)
      .subscribe(() => {
        this.fetchCart();
      });
  }
}
```

```
.medicine-container {
  margin-bottom: 20px;
}

.medicine-container h2 {
  font-size: 24px;
  margin-bottom: 10px;
}

.medicine-container form {
  display: flex;
  flex-direction: column;
}

.medicine-container label {
  font-weight: bold;
  margin-top: 10px;
}

.medicine-container input,
.medicine-container textarea {
  padding: 5px;
  margin-bottom: 10px;
}
```

```css
.medicine-container button[type="submit"] {
  background-color: #4caf50;
  color: white;
  padding: 8px 16px;
  border: none;
  cursor: pointer;
}

.medicine-container button[type="submit"]:hover {
  background-color: #45a049;
}

.medicine-list h2 {
  font-size: 24px;
  margin-bottom: 10px;
}

.medicine-list table {
  width: 100%;
  border-collapse: collapse;
}

.medicine-list th,
.medicine-list td {
  padding: 8px;
  text-align: left;
}

.medicine-list th {
  background-color: #f2f2f2;
  font-weight: bold;
}

.medicine-list tbody tr:nth-child(even) {
  background-color: #f2f2f2;
}

.medicine-list button {
  background-color: #f44336;
  color: white;
  padding: 5px 10px;
  border: none;
  cursor: pointer;
  margin-right: 5px;
```

```
}

.medicine-list button:hover {
  background-color: #e53935;
}

<div class="medicine-list">
  <h2>Medicine List</h2>
  <table>
    <!-- Table headers -->
    <thead>
      <tr>
        <th>Medicine Name</th>
        <th>Seller</th>
        <th>Description</th>
        <th>Price</th>
        <th>Action</th>
      </tr>
    </thead>
    <tbody>
      <ng-container *ngFor="let med of medicines">
        <tr *ngIf="med.editMode">
          <td>
            <input type="text" [(ngModel)]="med.name" name="name" required />
          </td>
          <td>
            <input
              type="text"
              [(ngModel)]="med.seller"
              name="seller"
              required
            />
          </td>
          <td>
            <textarea
              [(ngModel)]="med.description"
              name="description"
              required
            ></textarea>
          </td>
          <td>
            <input
              type="number"
              [(ngModel)]="med.price"
              name="price"
```

```
              required
            />
          </td>
          <td>
            <!-- Buttons for save and cancel -->
            <button (click)="updateMedicine(med)">Save</button>
            <button (click)="cancelEdit(med)">Cancel</button>
          </td>
        </tr>
        <tr *ngIf="!med.editMode">
          <td>{{ med.name }}</td>
          <td>{{ med.seller }}</td>
          <td>{{ med.description }}</td>
          <td>{{ med.price }}</td>
          <td>
            <!-- Buttons for edit and delete -->
            <button (click)="deleteMedicine(med.id)">Delete</button>
            <button (click)="toggleEditMode(med)">Edit</button>
          </td>
        </tr>
      </ng-container>
    </tbody>
  </table>
</div>

<style>
  /* Button Styles */
  .back-to-home-btn {
    display: inline-block;
    padding: 10px 20px;
    background-color: #007bff;
    color: #fff;
    text-decoration: none;
    border-radius: 4px;
    border: none;
    font-size: 16px;
    cursor: pointer;
  }
</style>
<a href="dashboard" class="back-to-home-btn">Back to Home</a>

<div class="signup-form">
  <h2>User Signup</h2>
  <form (ngSubmit)="signup()">
    <label>Name:</label>
```

```
    <input type="text" [(ngModel)]="user.name" name="name" required />
    <label>Email:</label>
    <input type="email" [(ngModel)]="user.email" name="email" required />
    <label>Phone:</label>
    <input type="text" [(ngModel)]="user.phone" name="phone" />
    <label>Password:</label>
    <input
      type="password"
      [(ngModel)]="user.password"
      name="password"
      required
    />
    <label>Address:</label>
    <textarea [(ngModel)]="user.address" name="address"></textarea>
    <button type="submit">Signup</button>
  </form>
  <div *ngIf="registrationSuccess" class="confirmation-message">
    <p>Registration successful!</p>
    <button class="btn-back" (click)="goToLoginPage()">Back to Login</button>
  </div>
</div>

import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css'],
})
export class UserComponent implements OnInit {
  user: any = {};
  registrationSuccess = false;

  constructor(private http: HttpClient, private router: Router) {}
  ngOnInit(): void {
    throw new Error('Method not implemented.');
  }

  signup() {
    this.http.post('http://localhost:8080/signup', this.user).subscribe(
      (response: any) => {
        console.log('Signup successful:', response);
      },
```

```
      (error: any) => {
        console.error('An error occurred during signup:', error);
      }
    );
    this.registrationSuccess = true;
  }
  goToLoginPage() {
    this.router.navigate(['/userlogin']);
  }
}

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { BehaviorSubject, Observable } from 'rxjs';

@Injectable({
  providedIn: 'root',
})
export class MedicineService {
  private apiUrl = 'http://localhost:8080';
  public search = new BehaviorSubject<string>('');

  constructor(private http: HttpClient) {}

  getMedicine(): Observable<any> {
    const url = `${this.apiUrl}/getMedicine`;
    return this.http.get(url);
  }

  addToCart(itemId: string): Observable<any> {
    const url = `${this.apiUrl}/addCart/${itemId}`;
    return this.http.post(url, {});
  }
}

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'filter',
})
export class MedicineFilterPipe implements PipeTransform {
  transform(value: any[], filterString: string, propName: string): any[] {
    const result: any = [];
    if (!value || filterString === '' || propName === '') {
      return value;
```

```
    }
    value.forEach((a: any) => {
      if (
        a[propName].trim().toLowerCase().includes(filterString.toLowerCase())
      ) {
        result.push(a);
      }
    });
    return result;
  }
}
```