KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

DEPARTMENT OF INFORMATION TECHNOLOGY

22ITL42 – WEB TECHNOLOGY LABORATORY

PROJECT REPORT

MATHAN M – 23ITR095

MOBILE CASE-ONLINE MOBILE CASE HUB

OBJECTIVES:

The objective of **CaseHub** is to offer an engaging and user-friendly platform where users can browse, search, and purchase stylish and protective mobile cases. Users can explore different case categories by model and brand, view detailed descriptions, leave ratings and reviews, and manage their shopping carts. The application ensures a seamless and secure checkout process with personalized recommendations and a responsive design.

TECHNOLOGY STACK

Backend Development:

- Programming Language: TypeScript
- Framework: Angular 16 (with Node.js/Express for API)
- Database: MongoDB (Non-relational)
- API Architecture: RESTful services

Frontend Development

- Core Technologies: HTML5, CSS3, SCSS
- UI Framework: Angular Material + Bootstrap 5
- State Management: NgRx (for complex state)
- Form Handling: Reactive Forms

MODULE DESCRIPTION:

- Home page
- Review Section
- Book Listening page
- Login/Signup page

ADVANTAGES:

- Enhanced User Experience
- Secure & Reliable
- Cost-Effective & Fast Development
- Competitive Advantage

SOURCE CODE:

home.component.html:

```
<div class="home-container">
  <!-- Hero Banner -->
  <div class="hero-banner">
    <div class="hero-content">
      <h1>Stylish & Protective Mobile Cases at Your Fingertips</h1>
      <div class="search-container">
        <input type="text"
               [(ngModel)]="searchQuery"
               (keyup.enter)="searchCases()"
               placeholder="Search mobile cases by phone model or brand..."
               class="search-input" />
        <button (click)="searchCases()" class="search-btn">
          <i class="fas fa-search"></i>
```

```
      </button>
    </div>
  </div>
</div>

<!-- Category Tabs -->
<div class="category-tabs">
  <div class="container">
    <div class="tabs-scroll">
      <button *ngFor="let category of categories"
              [class.active]="activeCategory === category"
              (click)="filterByCategory(category)"
              class="tab-btn">
        {{ category }}
      </button>
    </div>
  </div>
</div>

<!-- Featured Mobile Cases -->
<section class="featured-section" *ngIf="featuredCases.length > 0">
  <div class="container">
    <h2 class="section-title">Featured Mobile Cases</h2>
    <div class="row">
      <div *ngFor="let caseItem of featuredCases" class="col-md-4 mb-4">
        <div class="case-card featured">
          <div class="card-img">
            <img [src]="caseItem.image" [alt]="caseItem.name" />
            <div class="rating-badge">
              <i class="fas fa-star"></i> {{ caseItem.rating }}
            </div>
          </div>
          <div class="card-body">
            <h3>{{ caseItem.name }}</h3>
            <div class="brand">{{ caseItem.brand }}</div>
```

```
            <div class="details">
              <span class="price">
                <i class="fas fa-tag"></i> ₹{{ caseItem.price }}
              </span>
            </div>
            <button (click)="viewCase(caseItem._id)" class="btn view-btn">
              View Details
            </button>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>

<!-- All Mobile Cases -->
<section class="cases-section">
  <div class="container">
    <div class="section-header">
      <h2 class="section-title">All Mobile Cases</h2>
      <div class="sort-options">
        <select [(ngModel)]="sortOption" (change)="sortCases()" class="form-select">
          <option value="rating-desc">Rating: High to Low</option>
          <option value="price-asc">Price: Low to High</option>
          <option value="price-desc">Price: High to Low</option>
        </select>
      </div>
    </div>

    <div *ngIf="isLoading" class="loading-spinner">
      <div class="spinner-border text-primary" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
    </div>
```

```
<div *ngIf="!isLoading && filteredCases.length === 0" class="no-results">
  <i class="fas fa-box-open"></i>
  <h3>No cases found</h3>
  <p>Try adjusting your search or filters</p>
</div>

<div class="row">
  <div *ngFor="let caseItem of filteredCases" class="col-md-6 col-lg-4 mb-4">
    <div class="case-card">
      <div class="card-img">
        <img [src]="caseItem.image" [alt]="caseItem.name" />
        <div class="rating-badge">
          <i class="fas fa-star"></i> {{ caseItem.rating }}
        </div>
      </div>
      <div class="card-body">
        <h3>{{ caseItem.name }}</h3>
        <div class="brand">{{ caseItem.brand }}</div>
        <div class="details">
          <span class="price">
            <i class="fas fa-tag"></i> ₹{{ caseItem.price }}
          </span>
        </div>
        <button (click)="viewCase(caseItem._id)" class="btn view-btn">
          View Details
        </button>
      </div>
    </div>
  </div>
</div>
```

```
 <!-- Pagination -->
     <div *ngIf="totalPages > 1" class="pagination-container">
       <nav aria-label="Case pagination">
         <ul class="pagination">
           <li class="page-item" [class.disabled]="currentPage === 1">
             <a class="page-link" (click)="changePage(currentPage - 1)">Previous</a>
           </li>
           <li *ngFor="let page of getPageNumbers()" class="page-item" [class.active]="page
=== currentPage">
             <a class="page-link" (click)="changePage(page)">{{ page }}</a>
           </li>
           <li class="page-item" [class.disabled]="currentPage === totalPages">
             <a class="page-link" (click)="changePage(currentPage + 1)">Next</a>
           </li>
         </ul>
       </nav>
     </div>
   </div>
 </section>

 <!-- Promo Banner -->
 <div class="promo-banner">
   <div class="container">
     <div class="promo-content">
       <h2>Get 20% Off Your First Mobile Case!</h2>
       <p>Use code <strong>CASE20</strong> at checkout</p>
       <button class="btn order-now-btn">Shop Now</button>
     </div>
   </div>
 </div>
</div>
```

home.component.css:

```
.home-container {
  font-family: 'Poppins', sans-serif;




  .hero-banner {
    background: linear-gradient(rgba(0,0,0,0.5), url('/assets/images/hero-bg.jpg');
    background-size: cover;
    background-position: center;
    color: white;
    padding: 5rem 0;
    text-align: center;

    .hero-content {
      max-width: 800px;
      margin: 0 auto;

      h1 {
        font-size: 2.5rem;
        margin-bottom: 1.5rem;
      }
    }
  }

  .restaurant-card {
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 3px 10px rgba(0,0,0,0.1);
    transition: transform 0.3s ease;
```

```
    height: 100%;

    &:hover {
      transform: translateY(-5px);
    }

    .card-img {
      position: relative;
      height: 180px;
      overflow: hidden;

      img {
        width: 100%;
        height: 100%;
        object-fit: cover;
      }
    }
  }

  /* Add more styles for other elements */
}
```

auth.service.ts:

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

@Injectable({
  providedIn: 'root'
})
```

```
export class AuthService {
  private apiUrl = 'http://localhost:3000/api/auth';
  private currentUserSubject = new BehaviorSubject<any>(null);
  public currentUser = this.currentUserSubject.asObservable();

  constructor(private http: HttpClient, private router: Router) {
    const user = localStorage.getItem('currentUser');
    if (user) {
      this.currentUserSubject.next(JSON.parse(user));
    }
  }

  login(email: string, password: string): Observable<any> {
    return this.http.post(`${this.apiUrl}/login`, { email, password }).pipe(
      tap((response: any) => {
        if (response.token && response.user) {
          localStorage.setItem('currentUser', JSON.stringify(response.user));
          localStorage.setItem('token', response.token);
          this.currentUserSubject.next(response.user);
        }
      })
    );
  }

  signup(userData: any): Observable<any> {
    return this.http.post(`${this.apiUrl}/signup`, userData);
  }

  logout() {
    localStorage.removeItem('currentUser');
```

```
    localStorage.removeItem('token');
    this.currentUserSubject.next(null);
    this.router.navigate(['/login']);
  }

  getToken(): string | null {
    return localStorage.getItem('token');
  }

  isAuthenticated(): boolean {
    return !!this.getToken();
  }
}
```

auth.interceptor.ts:

```
import { Injectable } from '@angular/core';
import {
  HttpRequest,
  HttpHandler,
  HttpEvent,
  HttpInterceptor
} from '@angular/common/http';
import { Observable } from 'rxjs';
import { AuthService } from './auth.service';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  constructor(private authService: AuthService) {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler):
Observable<HttpEvent<unknown>> {
    const token = this.authService.getToken();
```

```
    if (token) {
      request = request.clone({
        setHeaders: {
          Authorization: `Bearer ${token}`
        }
      });
    }
    return next.handle(request);
  }
}
```

restaurant.service.ts:

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

interface Restaurant {
  _id: string;
  name: string;
  cuisine: string;
  deliveryTime: string;
  minOrder: number;
  rating: number;
  image: string;
}

interface MenuItem {
  _id: string;
  name: string;
  description: string;
```

```
  price: number;
  category: string;
  restaurantId: string;
  image: string;
}

@Injectable({
  providedIn: 'root'
})
export class RestaurantService {
  private apiUrl = 'http://localhost:3000/api/restaurants';
  private restaurantsSubject = new BehaviorSubject<Restaurant[]>([]);
  public restaurants$ = this.restaurantsSubject.asObservable();

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

  private loadRestaurants(): void {
    this.http.get<Restaurant[]>(this.apiUrl).subscribe({
      next: (restaurants) => this.restaurantsSubject.next(restaurants),
      error: (err) => console.error('Failed to load restaurants:', err)
    });
  }

  getRestaurantById(id: string): Observable<Restaurant> {
    return this.http.get<Restaurant>(`${this.apiUrl}/${id}`);
  }

  getMenu(restaurantId: string): Observable<MenuItem[]> {
```

```
    return this.http.get<MenuItem[]>(`${this.apiUrl}/${restaurantId}/menu`);
  }

  searchRestaurants(query: string): Observable<Restaurant[]> {
    return this.http.get<Restaurant[]>(`${this.apiUrl}/search?q=${query}`);
  }
}
```

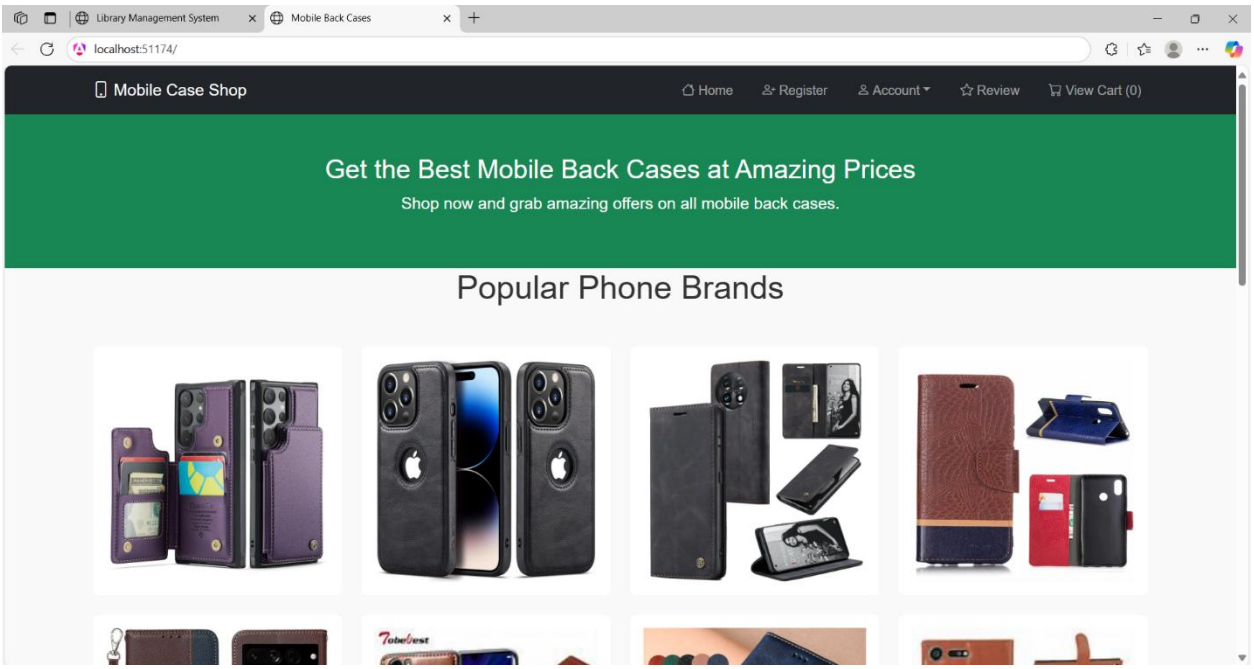
cart.service.ts:

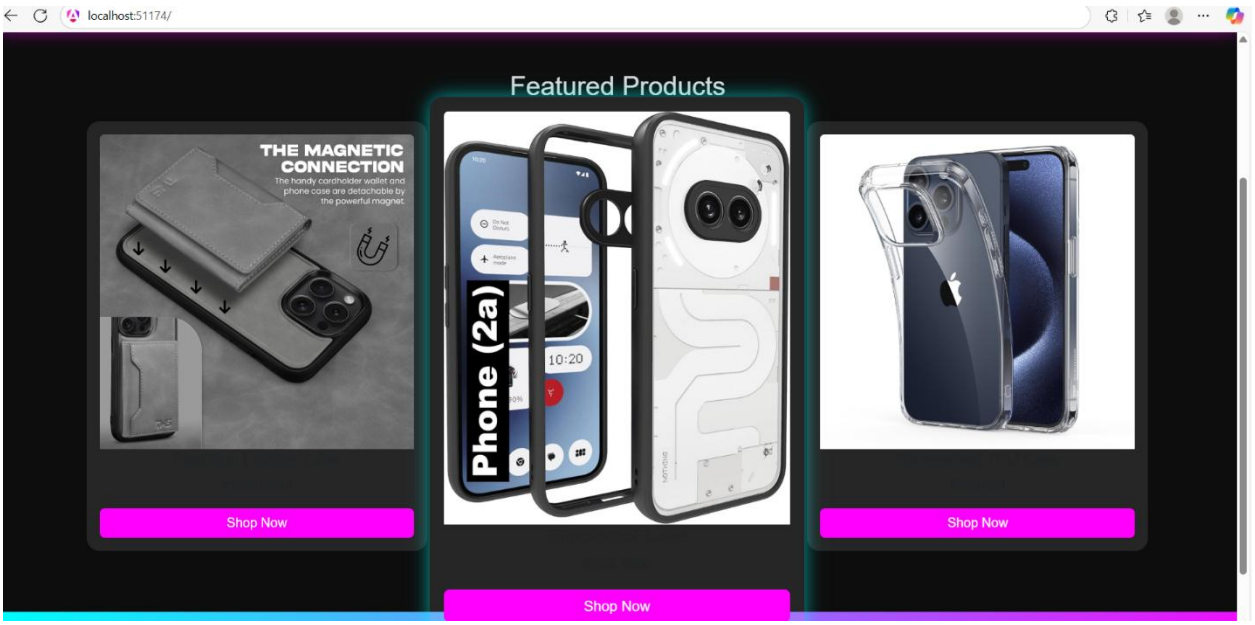
```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

interface CartItem {
  _id: string;
  name: string;
  price: number;
  quantity: number;
  restaurantId: string;
  image: string;
}

@Injectable({
  providedIn: 'root'
})
export class CartService {
  private cartItemsSubject = new BehaviorSubject<CartItem[]>([]);
  public cartItems$ = this.cartItemsSubject.asObservable();
  private cartTotalSubject = new BehaviorSubject<number>(0);
  public cartTotal$ = this.cartTotalSubject.asObservable();

  constructor() {
```

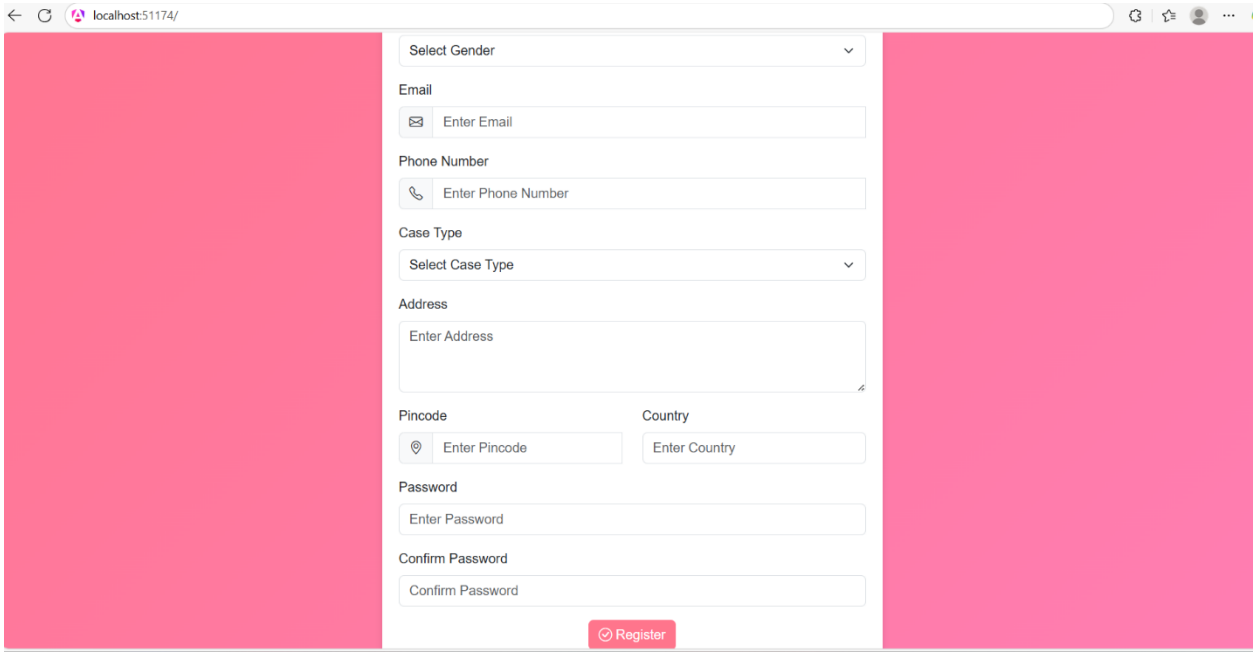
```
  const savedCart = localStorage.getItem('cart');
  if (savedCart) {
    this.cartItemsSubject.next(JSON.parse(savedCart));
    this.updateTotal();
  }
}

addToCart(item: CartItem): void {
  const currentItems = this.cartItemsSubject.value;
  const existingItem = currentItems.find(i => i._id === item._id);

  if (existingItem) {
    existingItem.quantity += 1;
  } else {
    currentItems.push({ ...item, quantity: 1 });
  }

  this.cartItemsSubject.next(currentItems);
  this.updateTotal();
  this.saveCart();
}

removeFromCart(itemId: string): void {
  const updatedItems = this.cartItemsSubject.value.filter(item => item._id !== itemId);
  this.cartItemsSubject.next(updatedItems);
  this.updateTotal();
  this.saveCart();
}

updateQuantity(itemId: string, quantity: number): void {
```

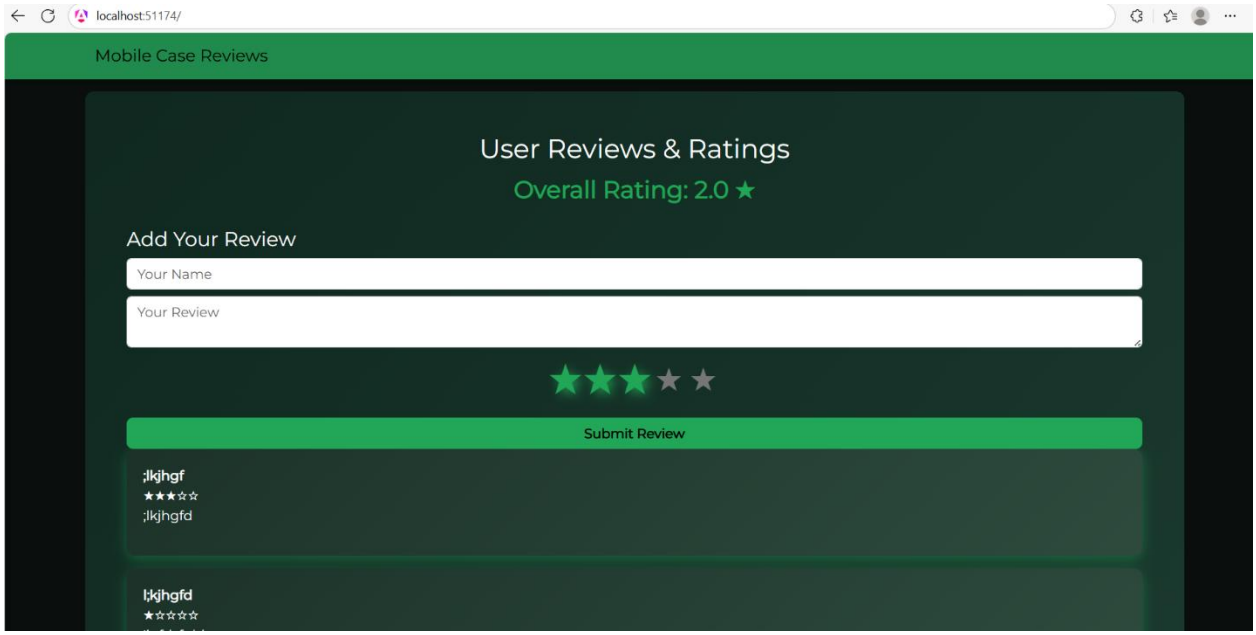
```
    const updatedItems = this.cartItemsSubject.value.map(item => {
      if (item._id === itemId) {
        return { ...item, quantity };
      }
      return item;
    });
    this.cartItemsSubject.next(updatedItems);
    this.updateTotal();
    this.saveCart();
  }

  clearCart(): void {
    this.cartItemsSubject.next([]);
    this.cartTotalSubject.next(0);
    localStorage.removeItem('cart');
  }

  private updateTotal(): void {
    const total = this.cartItemsSubject.value.reduce(
      (sum, item) => sum + (item.price * item.quantity),
      0
    );
    this.cartTotalSubject.next(total);
  }

  private saveCart(): void {
    localStorage.setItem('cart', JSON.stringify(this.cartItemsSubject.value));
  }
}
```

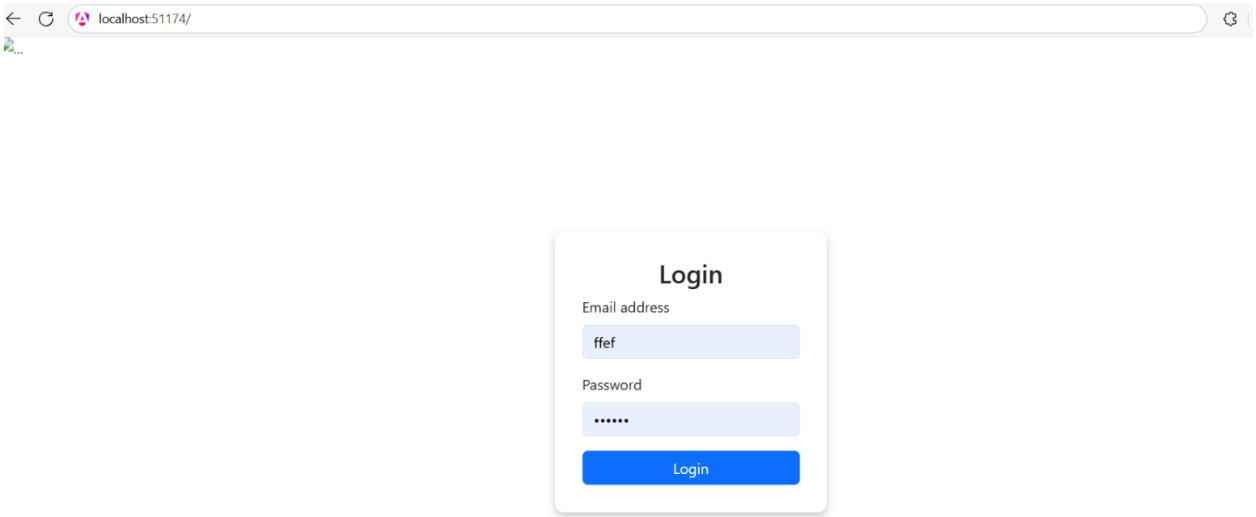
OUTPUT:


Library Management System
Mobile Back Cases
localhost:51174/
Mobile Case Shop
Home
Register
Account
Review
View Cart (0)
Get the Best Mobile Back Cases at Amazing Prices
Shop now and grab amazing offers on all mobile back cases.
Popular Phone Brands


localhost:51174/
Featured Products
THE MAGNETIC CONNECTION
The handy cardholder wallet and phone case are detachable by the powerful magnet.
Phone (2a)
10:20
Shop Now
Shop Now
Shop Now


localhost:51174/
Select Gender
Email
Enter Email
Phone Number
Enter Phone Number
Case Type
Select Case Type
Address
Enter Address
Pincode
Enter Pincode
Country
Enter Country
Password
Enter Password
Confirm Password
Confirm Password
Register


localhost:51174/
Mobile Case Reviews
User Reviews & Ratings
Overall Rating: 2.0 ★
Add Your Review
Your Name
Your Review
Submit Review
;lkjhgf
★★★☆☆
;lkjhgfd
l;kjhgfd
★☆☆☆☆


localhost:51174/
Login
Email address
ffef
Password
••••••
Login

CONCLUSION:

This **Online Mobile Case Shop** provides a modern e-commerce experience using Angular and Node.js with a responsive and user-centric interface. The project demonstrates robust front-end state management, efficient cart operations, secure user authentication, and seamless backend integration with MongoDB. Features like product filtering, review submission, and real-time updates make the platform scalable and engaging for users.

REFERENCES:

- *"Angular Up & Running" (2023) – Amazon*
- *"Node.js Design Patterns" (2020) – Packt Publishing*
- MongoDB Official Docs