```
================================
Step-1 : Angular Project Setup
================================

1) Download and install VsCode IDE

2) Setup Angular Environment

3) Create Angular Application

        $ ng new ashokit_ecomm_frontend

4) Run angular application

        $ cd ashokit_ecomm_frontend
        $ ng serve --open

Note: By default app-component will be loaded.

=================================================================
Step-2 : Retrieve Products From Backend and Display In Frontend
=================================================================

## 1) Create Product class to bind backend-app response in frontend-app

                        $ ng generate class common/Product


export class Product {

    constructor(
        public id: number,
        public name: string,
        public title: string,
        public description: string,
        public unitPrice: number,
        public imageUrl: string,
        public active: boolean,
        public unitsInStock: number,
        public dateCreated: Date,
        public lastUpdated: Date
    ){}
}

## 2) Configure HttpClientProvider in "app.config.ts" file

export const appConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes),
    provideHttpClient()
  ]
};



## 3) Create Service class to make HTTP Request to backend app

                $ ng generate service services/Product


```
```
```

~~~~
export class ProductService {

  private apiUrl = "http://localhost:8080/api/products";

  constructor(private httpClient: HttpClient) { }

  getProducts(): Observable<Product[]>{

    return this.httpClient.get<GetResponse>(this.apiUrl)
                   .pipe(map(response=> response._embedded.products));

  }
}
interface GetResponse{
    _embedded: {
      products: Product[];
    }
}
```

## 4) Create Product-List Component

                $ ng g c product-list

```
export class ProductListComponent implements OnInit {

  products: Product[] = [];

  constructor(private productService: ProductService) { }

  ngOnInit(): void {
    this.productService.getProducts().subscribe(data => {
      this.products = data;
    })
  }
}
```

### 5) Write presentation logic Product-List Component template file

```
<p *ngFor="let tempProduct of products">
    {{tempProduct.name }} :: {{tempProduct.unitPrice}}
</p>
```

### 6) Invoke Product-List Component from app-component using component selector

                <app-product-list></app-product-list>

### 7) Run the application and see products display


====================================================
Step-3 : Beautify Products Display in Table Format
====================================================

## 1) Add bootstap link in index.html file

<!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx"
crossorigin="anonymous">
~~~~

2) change app.component.html file to display product is container div like below

```
<div class="container mt-3 mb-3">
    <app-product-list></app-product-list>
</div>
```

3) Change product-list.component.html file to display products in table format

```
<table class="table">
    <thead class="table-dark">
        <tr>
            <th></th>
            <th>Name</th>
            <th>Price</th>
            <th>Units in Stock</th>
        </tr>
    </thead>
    <tbody>
        <tr *ngFor="let tempProduct of products">
            <td class="align-middle">
                <img src="{{tempProduct.imageUrl}}" height="50"/>
            </td>
            <td>{{tempProduct.name}}</td>
            <td>{{tempProduct.unitPrice}}</td>
            <td>{{tempProduct.unitsInStock}}</td>
        </tr>
    </tbody>
</table>
```

4) Add images folder under assets folder

5) Re-Start Angular app and check response in browser

```
===========================================
Step-4 : eCommerce Template Integration
===========================================
```

1) Install bootstap

```
  $ npm install bootstrap@5.2.0
```

2) Install FontAwesome

```
  $ npm install npm install @fortawesome/fontawesome-free
```

3) Verify installation entries in Node_Modules folder

4) Add Custom Styles in angular.json file

```

"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css",
  "node_modules/@fortawesome/fontawesome-free/css/all.min.css"
],

```

5) Add custom styles in styles.css file (copy and paste from our template)

6) modify app.component.html file to display menu and header part

```

<div class="page-wrapper">
    <!-- MENU SIDEBAR-->
    <aside class="menu-sidebar d-none d-lg-block">
        <div class="logo">
            <a href="#">
                <img src="assets/images/logo.png" alt="ashokit" class="img-responsive">
            </a>
        </div>
        <div class="menu-sidebar-content js-scrollbar1">
            <nav class="navbar-sidebar">
                <ul class="list-unstyled navbar-list">
                    <li>
                        <a href="#">Laptops</a>
                    </li>
                    <li>
                        <a href="#">Mobiles</a>
                    </li>
                    <li>
                        <a href="#">Shirts</a>
                    </li>
                    <li>
                        <a href="#">Beauty</a>
                    </li>
                </ul>
            </nav>
        </div>
    </aside>
    <!-- END MENU SIDEBAR-->
    <div class="page-container">
        <!-- HEADER DESKTOP-->
        <header class="header-desktop">
            <div class="section-content section-content-p30">
                <div class="container-fluid">
                    <div class="header-wrap">
                        <form class="form-header" onsubmit="return false;" method="GET">
                            <input class="au-input au-input-xl" type="text" name="search"
                                placeholder="Search for data ..." />
                            <button class="au-btn-submit" type="submit">
                                Search
                            </button>
                        </form>
                        <div class="cart-area d-n">
                            <a href="shopping-detail.html">
                                <div class="total">19.22 <span> 2</span> </div> <i class="fa fa-
shopping-cart"
                                    aria-hidden="true"></i>
                            </a>
                        </div>
                    </div>
                    <div class="account-wrap"></div>
                </div>
            </div>
        </header>
        <!-- END HEADER DESKTOP-->
        <!-- MAIN CONTENT-->
        <app-product-list></app-product-list>
    </div>

    <footer>
        <ul>
```

~~~
            <li><a href="#">About Us</a></li>
            <li><a href="#">Contact Us</a></li>
            <li><a href="#">Help</a></li>
        </ul>
    </footer>

```

## 7) Add Grid Support for product-list.html

```
<div class="main-content">
    <div class="section-content section-content-p30">
        <div class="row">
             <!-- loop over the collection of products -->
        <div *ngFor="let tempProduct of products" class="col-md-3">
            <div class="product-box">
                <img src="{{ tempProduct.imageUrl }}" class="img-responsive">
                <h1>{{ tempProduct.name }}</h1>
                <div class="price">{{ tempProduct.unitPrice }}</div>
                <a href="#" class="primary-btn">Add to cart</a>
            </div>
        </div>
        </div>
    </div>
</div>

```
===========================================
Step-5 : Filter Products with category
===========================================


## 1) Write method in service to fetch products based on category id

```
private apiUrl = "http://localhost:8080/api/products";

getProductsByCategory(theCategoryId: number): Observable<Product[]>{
    const searchUrl = `${this.apiUrl}/search/findByCategoryId?id=${theCategoryId}`;
    return this.httpClient.get<GetResponse>(searchUrl)
                          .pipe(map(response=> response._embedded.products));
}

```


## 2) Make changes in product-list component to get products based on category id

```
export class ProductListComponent implements OnInit {

  products: Product[] = [];
  currentCategoryId: number = 1;

  constructor(private productService: ProductService,
    private route: ActivatedRoute) { }

  ngOnInit(): void {
    this.route.paramMap.subscribe(() => {
      this.listProducts();
    })
  }

  listProducts() {
    const hasCategoryId: boolean = this.route.snapshot.paramMap.has('id');
~~~

```
    if (hasCategoryId) {
      // get category id and convert it into num type
      this.currentCategoryId = +this.route.snapshot.paramMap.get('id')!;
    } else {
      this.currentCategoryId = 1;
    }
    this.productService.getProductsByCategory(this.currentCategoryId).subscribe(data => {
      this.products = data;
    })
  }
}
```

3) Configure routerlink in app.component.html file (hardcoded links)

```
<li>
    <a routerLink="category/1" routerLinkActive="active-link">Laptops</a>
</li>
<li>
    <a routerLink="category/2" routerLinkActive="active-link">Mobiles</a>
</li>
```

4) Configure Routes in app.routes.ts file

```
export const routes: Routes = [

    {path: 'category/:id', component: ProductListComponent},
    {path: 'category', component: ProductListComponent},
    {path: 'products', component: ProductListComponent},
    {path: '', redirectTo: '/products', pathMatch: 'full'},
    {path: '**', redirectTo: '/products', pathMatch: 'full'},

];
```

5) make changes in app-component.html file to update router content using router-outlet.

 Remove : <app-product-list></app-product-list>

 Add : &lt;!-- MAIN CONTENT--&gt;
       <router-outlet></router-outlet>

6) Handle No Records found msg in product-list-component template

```

<div *ngIf="products.length==0" class="alert alert-warning col-md-12">
    No Records Found
</div>
```

==
Step-6 : Build Product Category Dynamic Menu
===

Kannan : build dynamic menu for categories

```
==========================================
Step-7 : Implement Search Functionality
==========================================

Gopal : Build search box functionality

======================================
Step-8 : Product Details Master View
======================================

## 1) Create new component ProuductDetails

    $ ng g c ProductDetails

## 2) Add routings for ProductDetails component like below

 {path: 'products/:id', component:ProductDetailsComponent},

## 3) Give routerLink for product name & image (change in productlist.component.html file)

<a routerLink="/products/{{tempProduct.id}}">
    <img src="{{ tempProduct.imageUrl }}" class="img-responsive">
</a>

<a routerLink="/products/{{tempProduct.id}}">
    <h1>{{ tempProduct.name }}</h1>
</a>

## 4) ProductDetailsComponent Template changes (just for testing component is loading or not)

<div class="detail-section">
    <div class="container-fluid">
        <p style="margin-top: 100px;">product-details works!</p>
    </div>
</div>

## 5) Service method to load product record based on product id


private apiUrl = "http://localhost:8080/api/products";

getProduct(theProductId:number): Observable<Product>{
    const productUrl = `${this.apiUrl}/${theProductId}`;
    return this.httpClient.get<Product>(productUrl);
}

## 6) Call Service method in component class using product id


export class ProductDetailsComponent implements OnInit {

  product!: Product;

  constructor(private productService: ProductService,
    private route: ActivatedRoute) {}

  ngOnInit(): void {
    this.route.paramMap.subscribe(() => {
      this.handleProductDetails();
    })
  }

  handleProductDetails() {
    // get id param value and call service method
    const theProductId: number = +this.route.snapshot.paramMap.get('id')!;
    this.productService.getProduct(theProductId).subscribe(data => {
      this.product = data;
    })

  }
}
```

7) ProductDetails Template change to display Product Master View

```
<div class="detail-section">
    <div class="container-fluid">
        <img src="{{product.imageUrl}}" class="detail-img">
        <h3>{{product.name}}</h3>
        <div class="price">{{product.unitPrice}}</div>
        <a href="#" class="primary-btn">Add To Cart</a>
        <hr />
        <h4>Description</h4>
        <p>{{product.description}}</p>
        <a routerLink="/products">Back To Product List Page</a>
    </div>
</div>
```

Note: Import RouterOutlet and RouterModel in component class.

```
======================================
Step-9 : Cart Functionality
======================================


======================================
Step-10 : Checkout Functionality
======================================
```

Step-1) When we click checkout button, open checkout form to capture customer details

a) customer info (name, email, phno)

b) shipping address

c) Click on 'Purchase' button
(send customer data + address details + order details to backend)

d) After step-2 completed, we will recieve order_tracking_num and razorpay_order_id

e) display razorpay payment popup (for payment processing) (angular with razorpay integration)

f) once payment success, display order_confirmation_page to customer

Step-2) backend functionality (new api)

SpringBoot with razorpay integration : https://www.youtube.com/live/t574YTo7HEw?si=aGlBSa0Np1NaMniQ

a) recieve customer data + address data + order details

b) If customer doesn't have account then create acc with random pwd and send in email

c) create razorpay order and get razorpay_order_id

d) insert order and order_items in db tables

```
        e) send order_tracking_num and razorpay_order_id to frontend


============================================
Step-10 : Login & Dashboard functionality
============================================

1) Develop backend microservie to handle Customer related functionalities

                a) Customer Registration

                b) Customer Login

                e) Forgot Pwd (similar to www.ashokit.in forgot pwd impl)

                d) Customer Dashboard (orders history)


2) Develop Frontend logic to handle above functionalities
```