```
Ui-app Git Repo: https://github.com/ashokitschool/ui-app.git

======================
eCommerce Application
======================

Project Architecture : https://www.youtube.com/watch?v=IbOMX4OYjSM

Backend : Spring Boot + Data JPA + Data REST + MySQL DB

Frontend : Angular 17v

====================
Application Modules
====================

Module-1) Products & Categories

Module-2) Products Search

Module-3) Cart

Module-4) Checkout

Module-5) Payment

Module-6) User Mgmt

Module-7) Orders

################################## Backend Development ##################################


===============================================================
Step-1) Create Spring Boot Application with below starters
===============================================================

a) spring-boot-starter-data-jpa

b) spring-boot-starter-data-rest

c) spring-boot-devtools

d) mysql-connector-j

=================================================
Step-2) Configure Data source properties
=================================================

spring.application.name=ashokit_ecomm_backend

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/jrtp26
spring.datasource.username=ashokit
spring.datasource.password=AshokIT@123

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.data.rest.base-path=/api

==================================
Step-3) Create Entity Classes
==================================

ProductCategory.java

Product.java

Note: One category can have multiple products

===========================================
Step-4) Create Rest Repository Interfaces
==========================================

ProductCategoryRepository.java

ProductRepository.java

================================================================
Step-5) Load Data into product_category and product tables
=================================================================

INSERT INTO product_category(category_name) VALUES ('Laptops');

        INSERT INTO product (sku, name, description, image_url, active, units_in_stock,
        unit_price, category_id, date_created)
        VALUES ('DELL-LAPTOP-1000', 'DELL - Laptop', 'Processor: Intel Core i5-1235U 12th Generation
(up to 4.40 GHz, 12MB 10 Cores)',
        'assets/images/products/laptops/dell-laptop-1000.png',1,100,19.00,1, NOW());

        INSERT INTO product (sku, name, description, image_url, active, units_in_stock,
        unit_price, category_id, date_created)
        VALUES ('HP-LAPTOP-1001', 'HP - Laptop', 'Processor: Intel Core i5-1235U 12th Generation (up
to 4.40 GHz, 12MB 10 Cores)',
        'assets/images/products/laptops/hp-laptop-1001.png',1,100,59,1, NOW());

        INSERT INTO product (sku, name, description, image_url, active, units_in_stock,
        unit_price, category_id, date_created)
        VALUES ('ACER-LAPTOP-1002', 'ACER - Laptop', 'Acer Aspire Lite 12th Gen Intel Core i5-1235U
Thin and Light Laptop ',
        'assets/images/products/laptops/acer-laptop-1002.png',1,100,49,1, NOW());

        INSERT INTO product (sku, name, description, image_url, active, units_in_stock,
        unit_price, category_id, date_created)
        VALUES ('LENOVO-LAPTOP-1003', 'Lenovo ThinkPad E14', 'Lenovo ThinkPad E14 AMD Ryzen 5',
        'assets/images/products/laptops/lenovo-laptop-1003.png',1,100,45.00,1, NOW());

INSERT INTO product_category(category_name) VALUES ('Mobiles');

        INSERT INTO product (sku, name, description, image_url, active, units_in_stock,
        unit_price, category_id, date_created)
        VALUES ('Apple-IPhone-1000', 'Apple-Iphone-Mobile', 'Apple iPhone 13 (128GB) - Green',
        'assets/images/products/mobiles/apple-mobile-1000.png',1,100,59,2, NOW());

        INSERT INTO product (sku, name, description, image_url, active, units_in_stock,
        unit_price, category_id, date_created)
        VALUES ('RED-MI-1001', 'Redmi 13C Mobile', 'Redmi 13C (Stardust Black, 6GB RAM, 128GB
Storage)',
        'assets/images/products/mobiles/redmi-mobile-1001.png',1,100,29,2, NOW());

        INSERT INTO product (sku, name, description, image_url, active, units_in_stock,unit_price,
category_id, date_created)
        VALUES ('SAMSUNG-Galaxy-1002', 'Samsung Mobile', 'SAMSUNG Galaxy F14 5G 6GB RAM 128GB
STORAGE',
        'assets/images/products/mobiles/samsung-mobile-1002.png',1,100,39,2, NOW());
        INSERT INTO product (sku, name, description, image_url, active, units_in_stock,unit_price,
category_id, date_created)
        VALUES ('POCO C65', 'POCO C65 Mobile', 'POCO C65 (Pastel Green 4GB RAM 128GB Storage)',
        'assets/images/products/mobiles/poco-mobile-1003.png',1,100,49,2, NOW());


===================================================
Step-6) Run the application and test in POSTMAN
===================================================

# URL to get all products
URL-1 : http://localhost:8080/api/products

# URL to get products based on product_id
URL-2 : http://localhost:8080/api/products/1

# URL to get all product categories
URL-3 : http://localhost:8080/api/product-category

# URL to get category based on category_id
URL-4 : http://localhost:8080/api/product-category/1

# URL to get products based on product_category_id
URL-5 : http://localhost:8080/api/product-category/1/products


################################## Frontend Development ##################################


================================
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 and Check

        $ cd ashokit_ecomm_frontend
        $ ng serve --open

5) Remove everything from app.component.html file and add your message

Note: It should reflect in web page


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

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

$ ng generate class Product

export class Product {

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

2) Create Service class to make HTTP call to backend app

        $ ng generate service services/Product

3) Configure HttpClientProvider in app.config.ts file

4) Create Product-List Component & access it in app-component using its selector

        $ ng g c product-list

5) Access Service Class Method in Product-List-Component  to fetch products using ngOnit

6) Write Presentation Logic to Display Products Data in template file


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

1) Add bootstap link in index.html file

===========================================
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 AppComponent.html file to display header, main-content and footer

7) Modify product-details-compontent template to display all products in grid format.


=========================================
Step - 5 : Search Products By Category
=========================================

1) Configure Routes in app.routes.ts file
```

```
        {path: 'category/:id', component:ProductListComponent},

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

        <a routerLink="category/1" routerLinkActive="active-link">Books</a>

3) Change ProductService to pass categoryId to backend app


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

0) Backend Change: Expose IDs for Entities by creating custom config class

1) Create ProductCategory class to map backend response

2) Create service method to get product categories

3) Create Product-Category-menu component.

4) Design menu in template

5) Remove static menu from app-component and invoke category-menu using selector.


=========================================
Step - 7 : Build Search Functionality
=========================================

1) Configure Router in app.module.ts file for search functionality

        {path: 'search/:keyword', component:ProductListComponent},

2) Create Search component

        $ ng generate component search

4) Remove static search form in app-component and invoke search-component using selector

5) design search component template

6) Handle Search functionality in product-list component

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

1) Create new component ProductDetails

        $ ng generate component ProductDetails

2) Add Router Link For ProductDetails

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

3) Give Router link for product-image and product-name in product-list-template

4) Write service method to get product based on product id

5) Invoke service method in product-details-component-ts file

6) Write presentation logic in template file to display product master view
```

```
=============================
*e-Commerce Project Status*
============================

1) Backend Development : Completed

2) Frontend project setup : Completed

3) Load products from backend and display in UI : Completed

4) Display product categories in side bar : Completed

5) Loading Products Based on Selected Category : Completed

6) Load products based on given search criteria : Completed

7) ProductDetailsMasterView : Completed

=========================================
Step - 9 : Shopping Cart
=========================================


1) Create CartItem class to represent cart data

        $ ng generate class CartItem

2) Create CartStatus Component

        $ ng generate component cart-status

3) Remove hard-coded cart-status details in app-component-template and
        Invoke cart-status Component from app-component

4) Create Cart-Service class to handle cart-related functionality

        addToCart() {}

        computeCartTotals() {}

5) Write a function to handle Add to Cart button in product-list-component

6) Write a function to handle Add to Cart button in product-master-view-component

7) subscribe to cart-values in cart-status-component

8) Access cart-status values in cart-status-template

=========================================
Step - 10 : Cart Details
=========================================

1) Create Cart-Details component

        $ ng g c cart-details

2) Configure Routing for Cart-Details

         { path: 'cart-details', component: CartDetailsComponent },

3) update cart-status-template to access cart-details-component

4) Access cart-items from cart-service and display them in cart-details page
```