Mini-Project 1: E-Commerce Website Using: Angular + Springboot + MySQL

References:

1. (92) 🔥How To Build a Full-Stack E-Commerce WebSite | Angular | Spring Boot | MySQL | 2024 | Simplilearn - YouTube

Will Work On:

- **Frontend Technology** : HTML | CSS | TYPESCRIPT | Angular
- **Backend Technology:** Java, Spring Framework, and Spring Boot
- **Communication Between Front-end and Back-end Technology**: Rest API
- **Database**: MySQL | Mongo db
- **To Connect Database** : JDBC | ORM < Hibernate or JPA | Spring Data JPA>

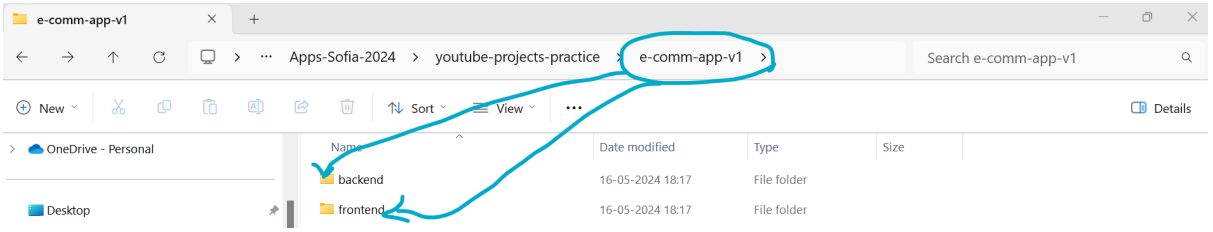
Work-space Path :

C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp

Let's create two folders:

1. Frontend folder
2. Backend folder

Back-end Part:



Login Module

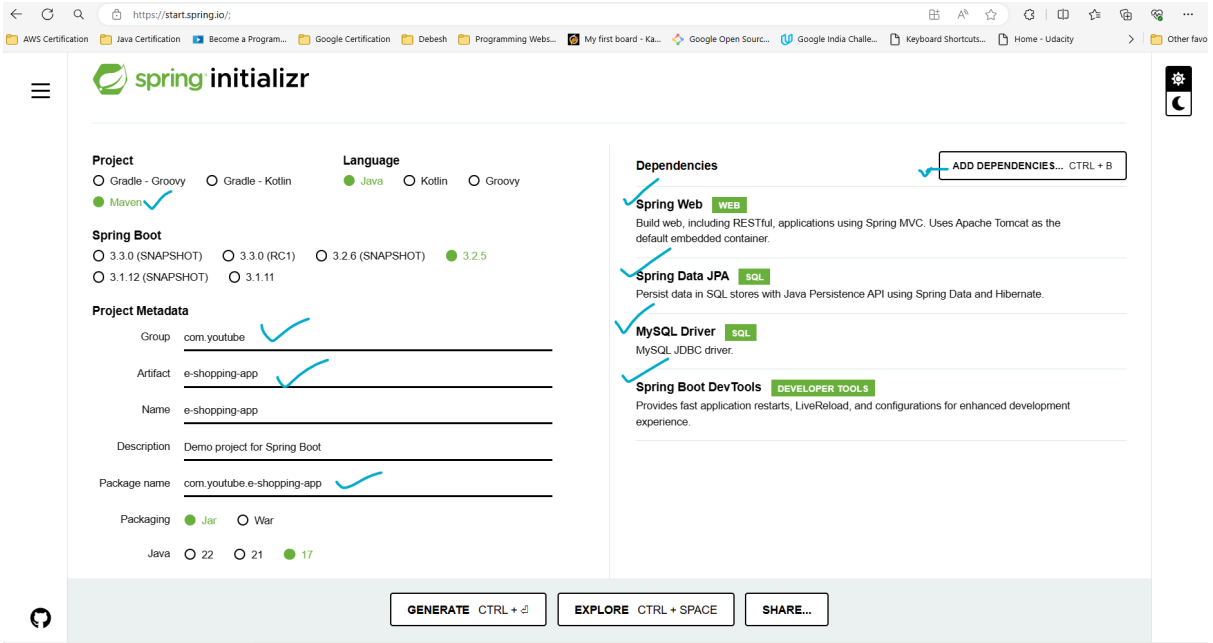
Product Module → add product ,delete product , update product and view product etc.

Order Module

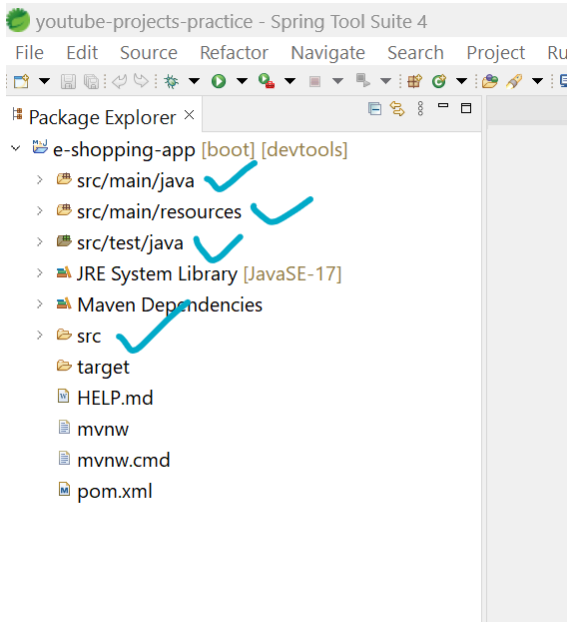
Account Module etc.

Create Spring Boot Project: e-shopping app

Go to: **spring initializr**



Keep the generated Maven folder inside the backend folder.
The setup is ready.



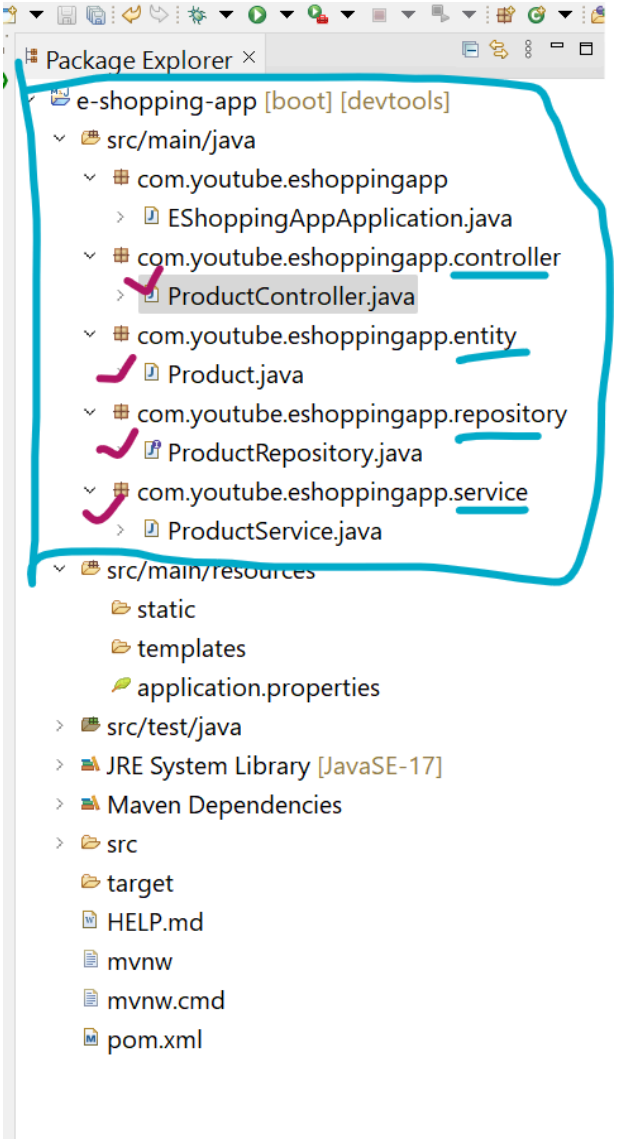
POM. XML → Project Object Model

```
<dependencies>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-jpa</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	</dependency>

	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-devtools</artifactId>
		<scope>runtime</scope>
		<optional>true</optional>
	</dependency>
	<dependency>
		<groupId>com.mysql</groupId>
		<artifactId>mysql-connector-j</artifactId>
		<scope>runtime</scope>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-test</artifactId>
		<scope>test</scope>
```

```
        </dependency>
    </dependencies>
```

DAO layer: Data Access ….

File Structure 👍



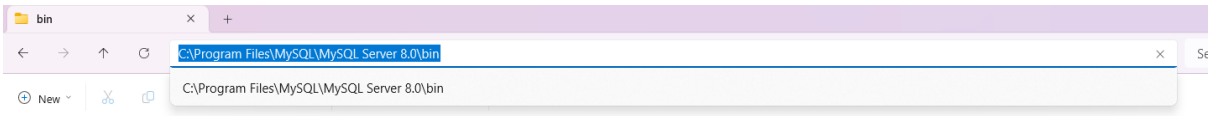
Now the Product setup is ready.
But where do we store the data? → In database

MySQL: Create Database: capstone

Way-1: Establish MySQL Connectivity via Command Line: A Quick Guide

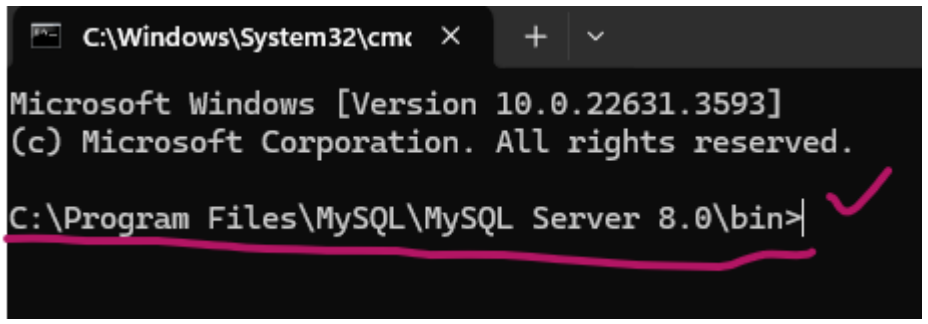
Step1: Access your MySQL server.

Step2: Navigate to the desired path.



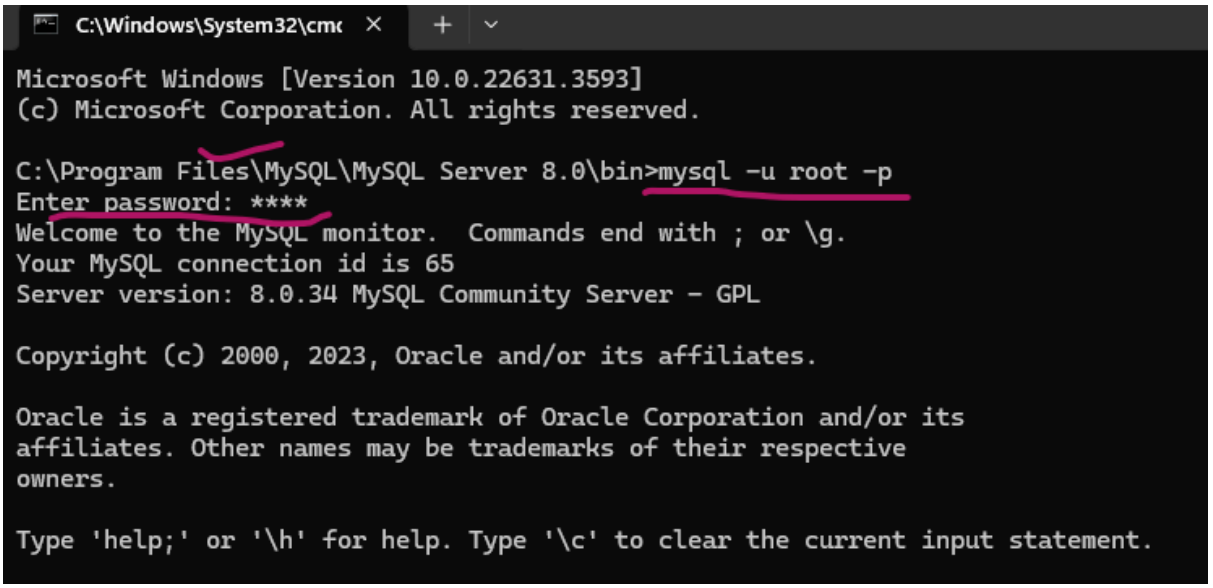
Step-3:Open the Command Prompt by typing "cmd".

Step-4:The Command Prompt will launch with the chosen path.



Step-5: Input the command: `mysql -u root -p` (replace `root` with your username if different).

Step-6 : You'll be prompted to enter your MySQL password. (e.g., password: `root`)



Step-7: Test the connection with a query like `show databases;`

Way-2: Connect Using MySQL Workbench(Optional):

A graphical tool provided by MySQL to connect MySQL SERVER.

- **Open MySQL Workbench → 3306**
- **Write the command to create the database and others**

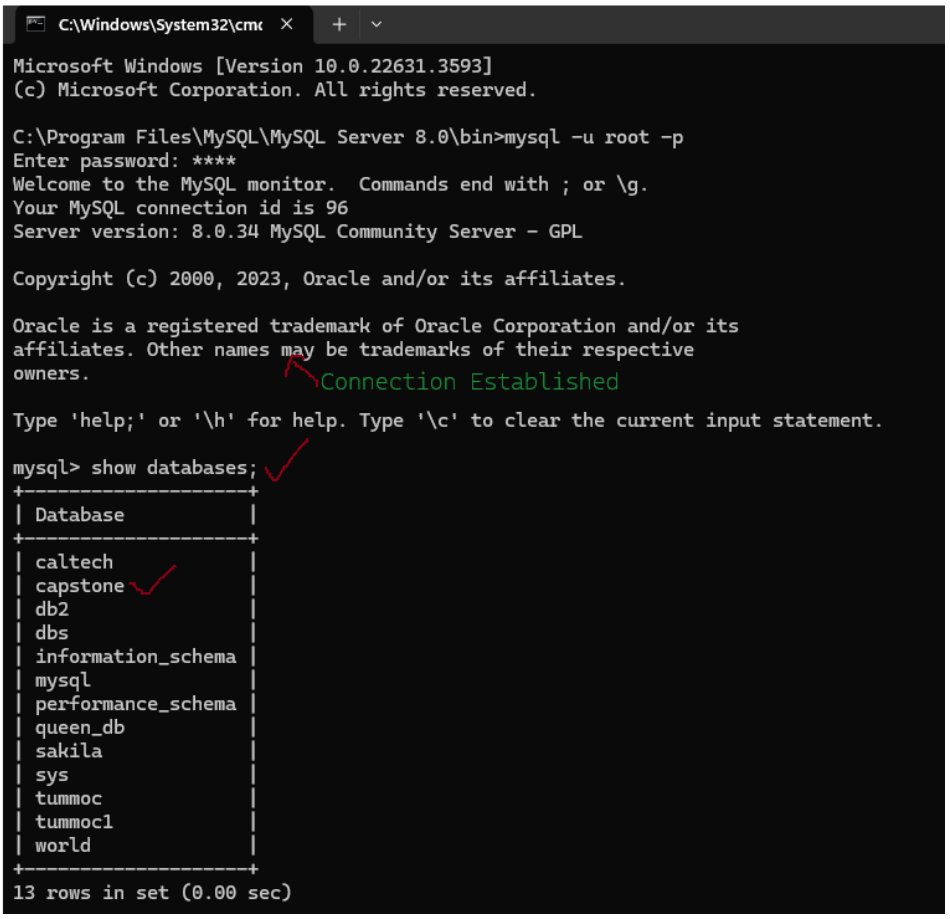
```
create database capstone;
use capstone;
show tables;
```

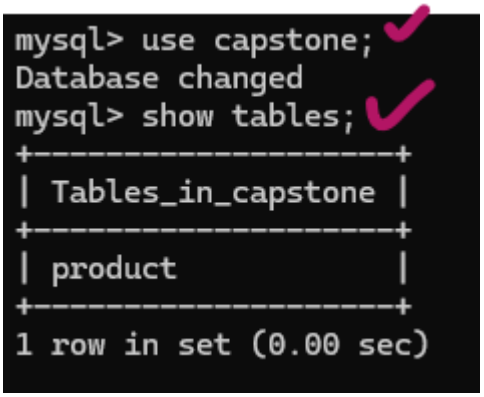
Establish a Connection To the Database from Spring Boot

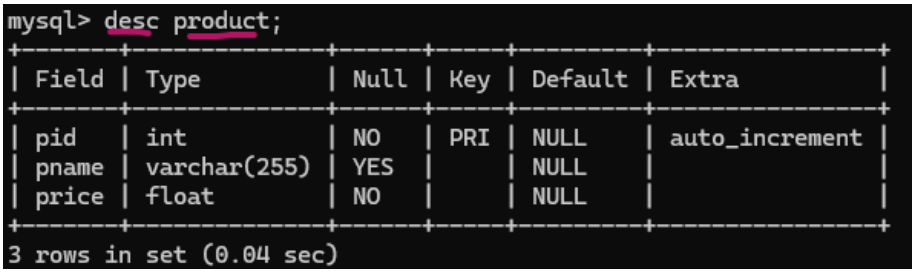
Open: application.properties

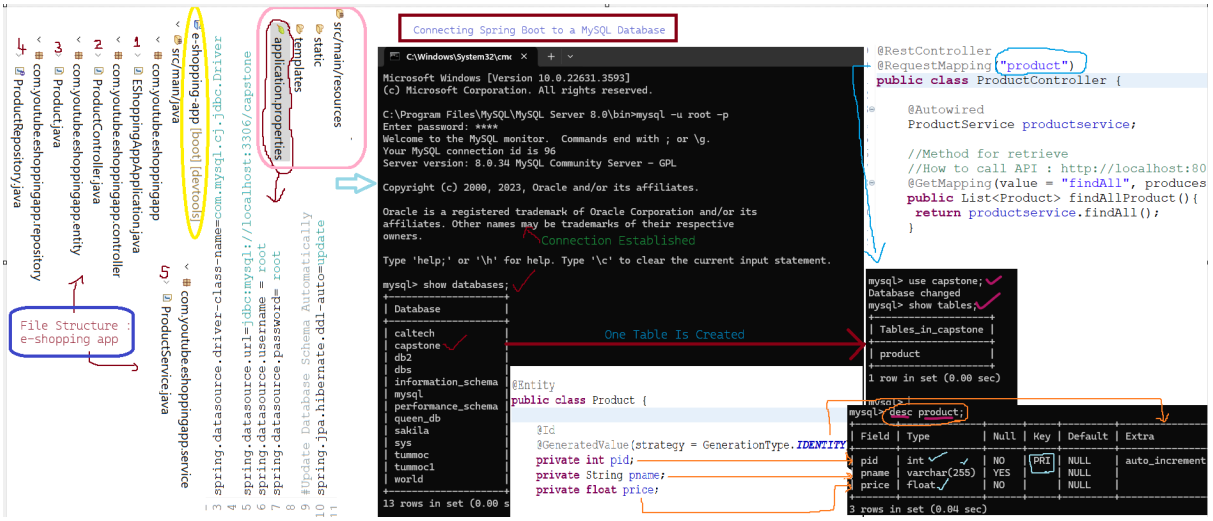
Then Write the following :

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/capstone
spring.datasource.username = root
spring.datasource.password= root
#Update Database Schema Automatically
spring.jpa.hibernate.ddl-auto=update
```


C:\Windows\System32\cmd
Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 96
Server version: 8.0.34 MySQL Community Server - GPL
Copyright (c) 2000, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Connection Established
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
Database
caltech
capstone
db2
dbs
information_schema
mysql
performance_schema
queen_db
sakila
sys
tummoc
tummoc1
world
13 rows in set (0.00 sec)


mysql> use capstone;
Database changed
mysql> show tables;
Tables_in_capstone
product
1 row in set (0.00 sec)


mysql> desc product;
Field | Type | Null | Key | Default | Extra
pid | int | NO | PRI | NULL | auto_increment
pname | varchar(255) | YES | | NULL |
price | float | NO | | NULL |
3 rows in set (0.04 sec)



Now the simple **backend part** is ready with the help of **spring-boot and MySQL database.**

Note:

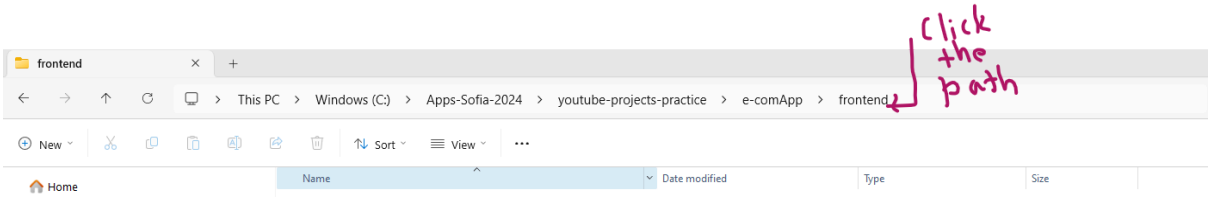
In Entity Class only need getter() and setters(). No need for any ***constructors() and toString() method.***

Front-end Part:

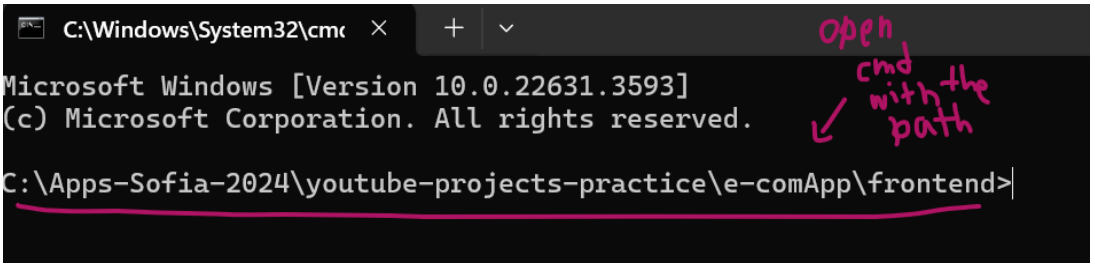
Now coming to the front-end part of technology.

In front-end technology, we are using → **Angular**
And for angular, we need → **node.js**

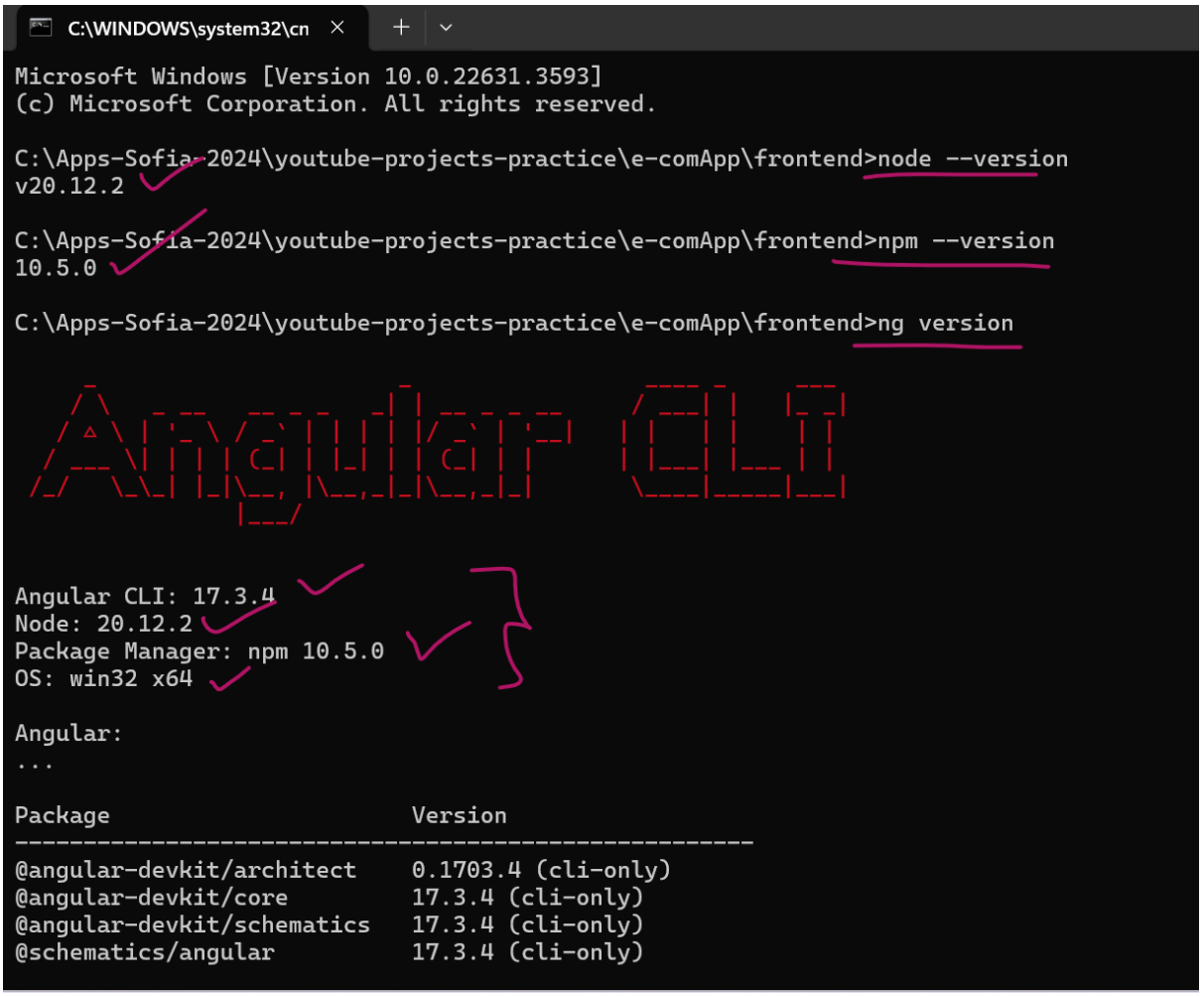
Go to the path. Navigate it.



Write **cmd** there.



Let's know the **version** of: **node | npm | ng**



Create an Angular Project:

Now we are going to create an **Angular Project**.

Commands We need to **execute now**:

ng new front-end-app - - no - standalone → Create app.module.ts file

```
C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp\frontend>ng new front-end-app  --no-standalone
? Which stylesheet format would you like to use? CSS                [ https://developer.mozilla.org/docs/Web/CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? Yes
CREATE front-end-app/angular.json (3136 bytes)
```

cd front-end-app → inside the project folder

```
C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp\frontend>cd  front-end-app

C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp\frontend\front-end-app>
```

ng g c product → Create component product

```
C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp\frontend\front-end-app>ng g c product
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

    ng analytics disable

Global setting: enabled
Local setting: enabled
Effective status: enabled
CREATE src/app/product/product.component.html (23 bytes)
CREATE src/app/product/product.component.spec.ts (631 bytes)
CREATE src/app/product/product.component.ts (213 bytes)
CREATE src/app/product/product.component.css (0 bytes)
UPDATE src/app/app.module.ts (555 bytes)
```

ng g class product → A model class. To map the java bean/entity class; we need a model class

```
C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp\frontend\front-end-app>ng g class product
CREATE src/app/product.spec.ts (165 bytes)
CREATE src/app/product.ts (27 bytes)
```

ng g s product → Create service product

```
C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp\frontend\front-end-app>ng g s product
CREATE src/app/product.service.spec.ts (378 bytes)
CREATE src/app/product.service.ts (145 bytes)
```

Using IDE : VSCode

```
C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp\frontend\front-end-app>code .
```

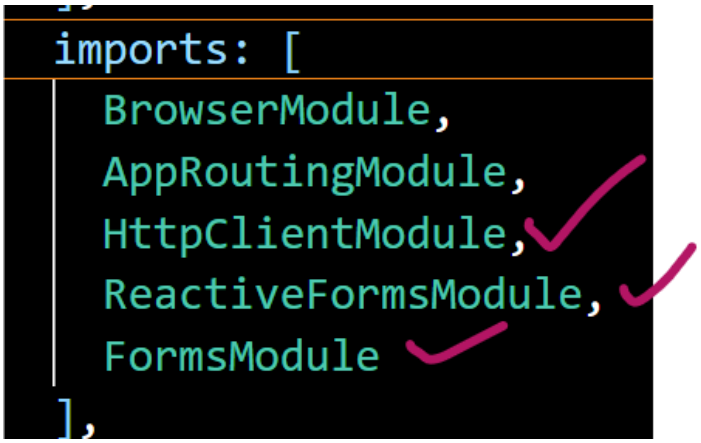
Routing: is the concept which has to navigating from one page to another page

File-1 app.module.ts :

We need a few modules that we import in Angular.

1. HttpClientModule → This module is called the **REST API**
2. RactiveFormsModule, FormsModule → These two are used to **create the forms**

```
import { HttpClientModule } from '@angular/common/http';
import { ReactiveFormsModule, FormsModule } from '@angular/forms';
```



This part is ready now.

File-2 product.ts :: The Model class

This class is used to map to JSON data.

src\app\product.ts

```
export class Product {
```

```
//Used to map to the json data , which is exactly like
java entity class

constructor(
    public pid: number,
    public pname: string,
    public price:number
){}

}
```

File-3 product.service.ts

Now coming to the service class:

src\app\product.service.ts

HttpClient is a **predefined API** provided by Angular: Which we have to call **REST API**

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Product } from './product';

//This service file work to interact the front-end with
back-end

@Injectable({
  providedIn: 'root'
})
export class ProductService {

  constructor(public http:HttpClient) { }
```

```
  //For Retrieving Data
  loadProduct():Observable<Product[]>{
    return
this.http.get<Product[]>("http://localhost:8080/product
/findAll");
  }

  //For Storing Data
  storeProduct(product:any):Observable<string>{
    return
this.http.post("http://localhost:8080/product/store",pr
oduct,{responseType:'text'})
  }
}
```

File-4: product.component.ts

src\app\product\product.component.ts

Import the service file[i.e. ProductService] in constructor.

```
import { Component, OnInit } from '@angular/core';
import { ProductService } from '../product.service';
import { Product } from '../product';
import { FormControl, FormGroup } from
'@angular/forms';

@Component({
  selector: 'app-product',
  templateUrl: './product.component.html',
  styleUrl: './product.component.css'
})
export class ProductComponent implements OnInit{
```

```
  //Create an Array of Product Type
  products: Array<Product> = [];

  constructor(public ps: ProductService){}

  ngOnInit(): void {
    //It called automatically when the component was
loaded
  }

//For store the product we should create a form group
for template
 productRef = new FormGroup({
  pid : new FormControl(),
  pname : new FormControl(),
  price : new FormControl()
 });


  loadProduct():void{
    this.ps.loadProduct().subscribe({
      next:(result:any) =>{
        this.products=result;   //First Method : next()
      },
      error:(error:any) =>{
        console.log(error);     //Second Method:
error()
      },
      complete:() => {          //Third Method:
complete()
        console.log("Done!!!")
      }
    }); //End of subscribe()
```

```
  }

  msg:string="";

  storeProduct():void{
   let product = this.productRef.value;
   this.ps.storeProduct(product).subscribe({

    next:(result)=> {           //First Method: next()
      this.msg = result;
    },
    error:(error:any) => {      //Second Method: error()
      console.log (error);
    },

    complete:()=>{             //Third Method :
complete()
      console.log("Done!!!");
    }
   }); //End of subscribe()


  }

}
```

File-5 product.component.html

src\app\product\product.component.html

Basic Form [Is Ready] :

```
<div>
    <!--For Input: -->

<form [formGroup]="productRef"
(ngSubmit)="storeProduct()"> <!--Bind the component-->
    <label>Product Name:</label>
    <input type="text" formControlName="pname">
    <br/>
    <label>Price:</label>
    <input type="number" formControlName="price">
    <br/>
    <br/>
    <input type="submit" value="store product">
    <input type="reset"  value="reset">
</form>

<br/>
<br/>
<!--For Displaying Information : Table Format-->
<table border="1px">
    <tr>
       <th>Product ID</th>
       <th>Product Name</th>
       <th>Price</th>
    </tr>

    <tr *ngFor="let product of products">
       <td>{{product.pid}}</td>
       <td>{{product.pname}}</td>
```

```
        <td>{{product.price}}</td>
    </tr>

</table>
</div>
```

Updated Form:

```
<div class="container">
    <!--For Input: -->
    <h2>Add Product</h2>

<form [formGroup]="productRef"
(ngSubmit)="storeProduct()"> <!--Bind the component-->
    <label>Product Name:</label>
    <input type="text" formControlName="pname"
class="form-control">
    <br/>
    <label>Price:</label>
    <input type="number" formControlName="price"
class="form-control">
    <br/>
    <br/>
    <input type="submit" value="Store Product"
class="btn btn-success">
    <input type="reset"  value="Reset" class="btn
btn-info">
</form>

<br/>
<br/>
<!--For Displaying Information : Table Format-->
<table border="1px" class="table">
```

```
    <tr>
        <th>Product ID</th>
        <th>Product Name</th>
        <th>Price</th>
    </tr>

    <tr *ngFor="let product of products">
        <td>{{product.pid}}</td>
        <td>{{product.pname}}</td>
        <td>{{product.price}}</td>
    </tr>

</table>
</div>
```

File-6 app.component.html

src\app\app.component.html
Except <router-outlet/> remove all and add <app-product> there.

```
<app-product></app-product>
<router-outlet />
```

Run the project: Open Command Prompt
ng serve -o

```
C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp\frontend\front-end-app>ng serve -o
```

It will open the project in the browser.

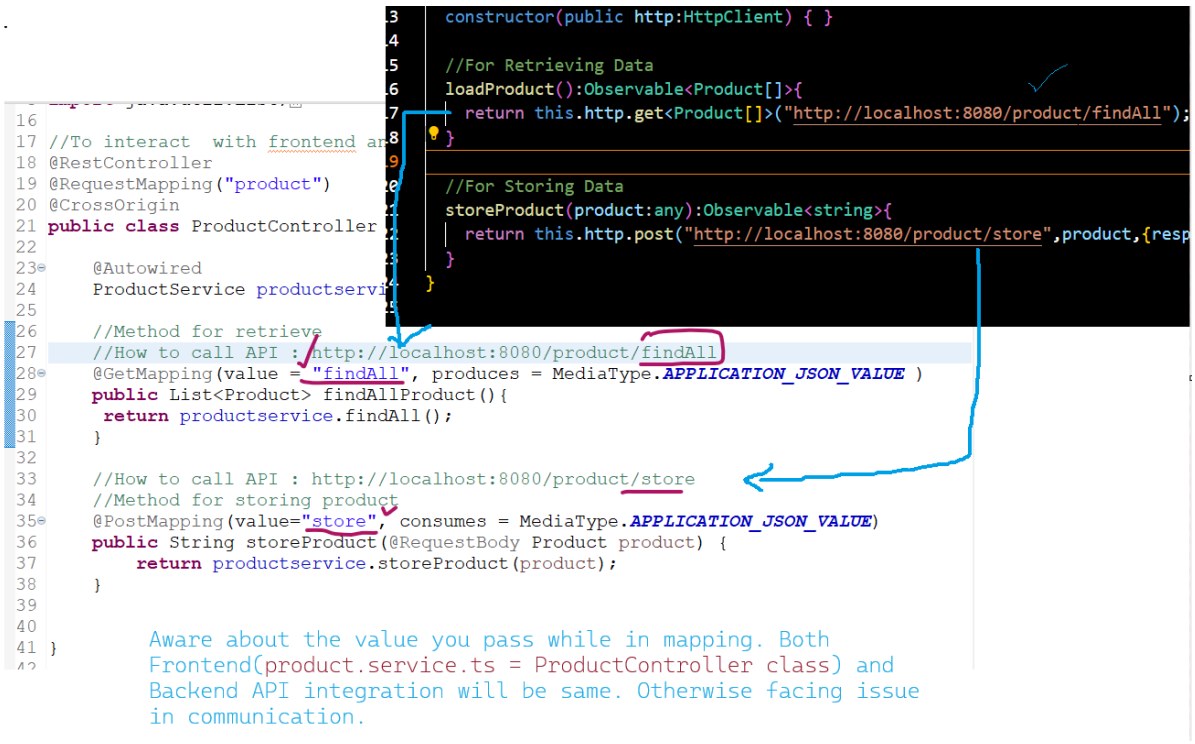
CORS : ISSUE [Resolved]

To avoid the CORS Issue , in the backend side we do add an annotation in the file :

ProductController.java

```java
import com.youtube.eshoppingapp.entity.Product;
import com.youtube.eshoppingapp.service.ProductService;

//To interact  with frontend and backend


@RestController
@RequestMapping("product")
@CrossOrigin //It helps to communicate with frontend and bac
public class ProductController {

    @Autowired
    ProductService productservice;

```

API:ISSUE [Resolved]



File–7 Index.html

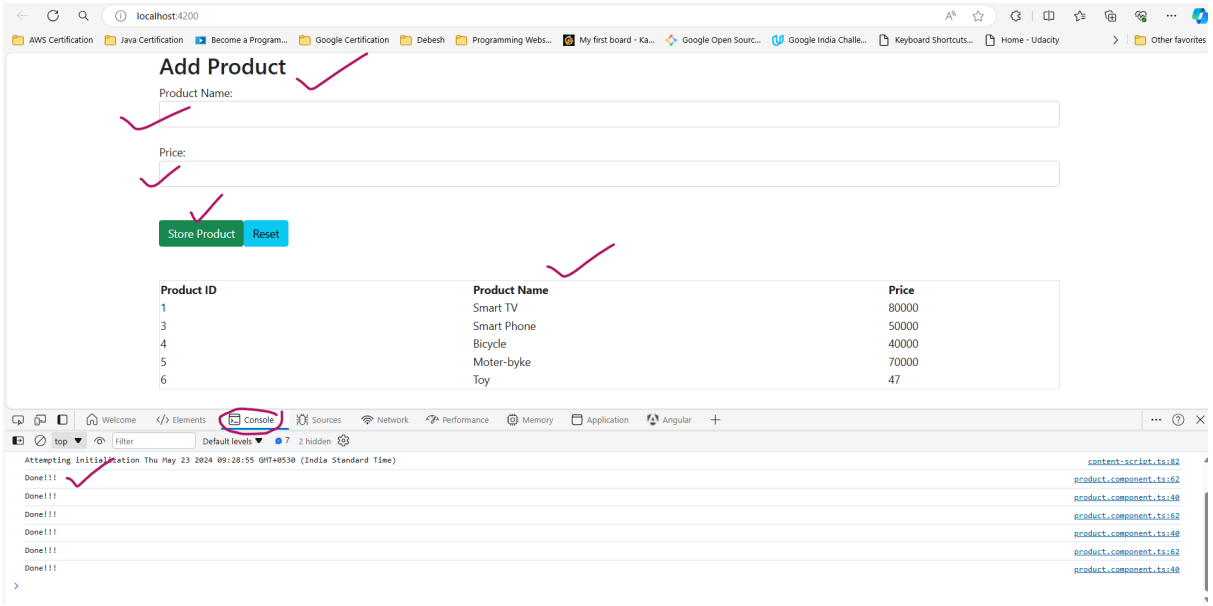
Add Bootstrap Feature 👍

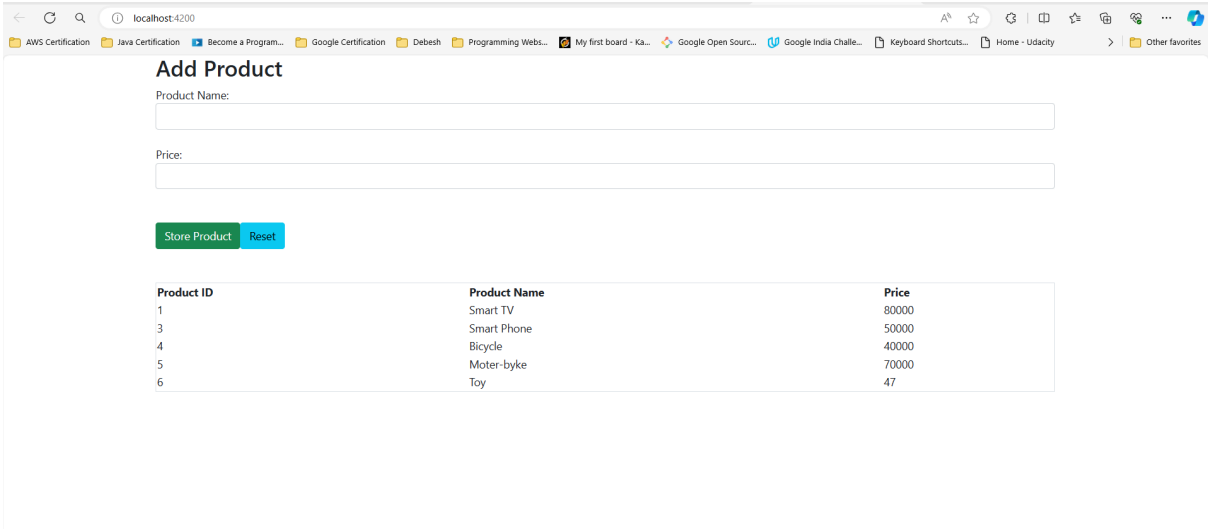
Add this css link of bootstrap inside <head> section.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
```

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>FrontEndApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Result:


localhost:4200
Add Product
Product Name:
Price:
Store Product
Reset
Product ID
Product Name
Price
1
Smart TV
80000
3
Smart Phone
50000
4
Bicycle
40000
5
Moter-byke
70000
6
Toy
47
Welcome
Elements
Console
Sources
Network
Performance
Memory
Application
Angular
Attempting initialization Thu May 23 2024 09:28:55 GMT+0530 (India Standard Time)
content-script.ts:82
Done!!!
product.component.ts:62
product.component.ts:40
product.component.ts:62
product.component.ts:40
product.component.ts:62
product.component.ts:40


localhost:4200
Add Product
Product Name:
Price:
Store Product
Reset
Product ID
Product Name
Price
1
Smart TV
80000
3
Smart Phone
50000
4
Bicycle
40000
5
Moter-byke
70000
6
Toy
47

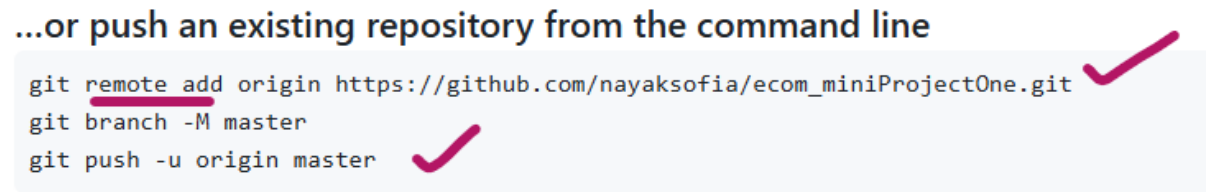
```
mysql> select * from product;
+-----+---------------+--------+
| pid | pname         | price  |
+-----+---------------+--------+
|   1 | Smart TV      |  80000 |
|   3 | Smart Phone   |  50000 |
|   4 | Bicycle       |  40000 |
|   5 | Moter-byke    |  70000 |
|   6 | Toy           |     47 |
+-----+---------------+--------+
5 rows in set (0.00 sec)
```

Git And Github: Mini-Project-1

Command to remove .git from frontend application:

rmdir /s /q .git

Create a repository in GitHub:




```
C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp>git remote add origin https://github.com/nayaksofia/ecom_miniProjectOne.git
C:\Apps-Sofia-2024\youtube-projects-practice\e-comApp>git push -u origin master
```