

UES20CS352 – OOAD

MVC Assignment

Name: Vishwa Mehul Mehta

SRN: PES2UG20CS389

Sec: F

Date: 10-04-2023

1.

a. MVC Architecture Pattern:

MVC (Model-View-Controller) is a pattern in software design commonly used to implement user interfaces, data, and controlling logic. It emphasizes a separation between the software's business logic and display. This "separation of concerns" provides for a better division of labor and improved maintenance. Some other design patterns are based on MVC, such as MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter), and MVW (Model-View-Whatever).

The three parts of the MVC software-design pattern can be described as follows:

1. Model: Manages data and business logic.
2. View: Handles layout and display.
3. Controller: Routes commands to the model and view parts.

The Model:

The model defines what data the app should contain. If the state of this data changes, then the model will usually notify the view (so the display can

change as needed) and sometimes the controller (if different logic is needed to control the updated view).

Going back to our shopping list app, the model would specify what data the list items should contain – item, price, etc. – and what list items are already present.

The View:

The view defines how the app's data should be displayed.

In our shopping list app, the view would define how the list is presented to the user, and receive the data to display from the model.

The Controller:

The controller contains logic that updates the model and/or view in response to input from the users of the app.

So for example, our shopping list could have input forms and buttons that allow us to add or delete items. These actions require the model to be updated, so the input is sent to the controller, which then manipulates the model as appropriate, which then sends updated data to the view.

You might however also want to just update the view to display the data in a different format, e.g., change the item order to alphabetical, or lowest to highest price. In this case the controller could handle this directly without needing to update the model.

b. Advantages of MVC pattern:

i. An improved development process:

By using MVC, you can develop in parallel and in a rapid manner. To develop the business logic of an MVC web application, one programmer can work on the view and the other on the controller. Therefore, applications developed using the MVC model can be completed three times faster than those developed using other techniques.

ii. Ability To Provide Multiple Views:

A model can have multiple views using the MVC Model. MVC development is a great solution for today's increasing demand for new methods of accessing your applications. The approach also limits the amount of code duplication since the display is separated from the data and business logic.

iii. Support For Asynchronous Technique:

MVC architecture can also be integrated with JavaScript Framework. In other words, MVC applications will be able to work with PDF files, site-specific browsers, and andns. The MVC framework also supports an asynchronous technique that helps developers develop applications that load very quickly.

iv. The Modification Does Not Affect The Entire Model:

For any web application, the user interface tends to change more frequently than even the business rules of the .net development company. It is obvious that you make frequeThe MVC framework also supports an asynchronous technique that helps developers develop applications that load very quickly.or tablets. Moreover, Adding a new type of view is very easy in the MVC pattern because the Model part does not depend on the views part.

Therefore, any changes in the Model will not affect the entire architecture.

v. MVC Model Returns The Data Without Formatting:

MVC pattern returns data without applying any formatting. Hence, the same components can be used and called for use with any interface. For example, any kind of data can be. The MVC framework also supports an asynchronous technique that helps developers develop applications that load very quickly.

vi. SEO Friendly Development Platform:

MVC platform supports the development of SEO friendly web pages or web applications. Using this platform, it is very easy to develop SEO-friendly URLs to generate more visits. This technique allows developers to develop an application that loads rapidly. MVC also supports asynchronous development. Moreover, Scripting languages like JavaScript and [jQuery](#) can be integrated with MVC to develop feature-rich web applications.

Thus, the MVC design pattern is surely a great approach to building software applications. The MVC framework is easy to implement as it offers above given numerous advantages asynchronous technology, MVC also supports fast, which helps developers develop a fast loading application. Above all, its power to manage multiple views makes MVC the best architecture pattern for developing web applications.

As a result, today organizations are looking for the software development of web applications based on MVC architecture for cost and time benefits. There are many web development companies providing MVC development services to develop web applications that satisfy every requirement of the

clients. Profound Edutech is one such software development institute that provides the most desired output to its students by offering fast and highly interactive web applications using MVC 6 development architecture. Contact us today.

c. Features of Spring MVC:

Spring's web module includes many unique web support features:

- Clear separation of roles. Each role – controller, validator, command object, form object, model object, DispatcherServlet, handler mapping, view resolver, and so on – can be fulfilled by a specialized object.
- Powerful and straightforward configuration of both framework and application classes as JavaBeans. This configuration capability includes easy referencing across contexts, such as from web controllers to business objects and validators.
- Adaptability, non-intrusiveness, and flexibility. Define any controller method signature you need, possibly using one of the parameter annotations (such as `@RequestParam`, `@RequestHeader`, `@PathVariable`, and more) for a given scenario.
- Reusable business code, no need for duplication. Use existing business objects as command or form objects instead of mirroring them to extend a particular framework base class.
- Customizable binding and validation. Type mismatches as application-level validation errors that keep the offending value, localized

date and number binding, and so on instead of String-only form objects with manual parsing and conversion to business objects.

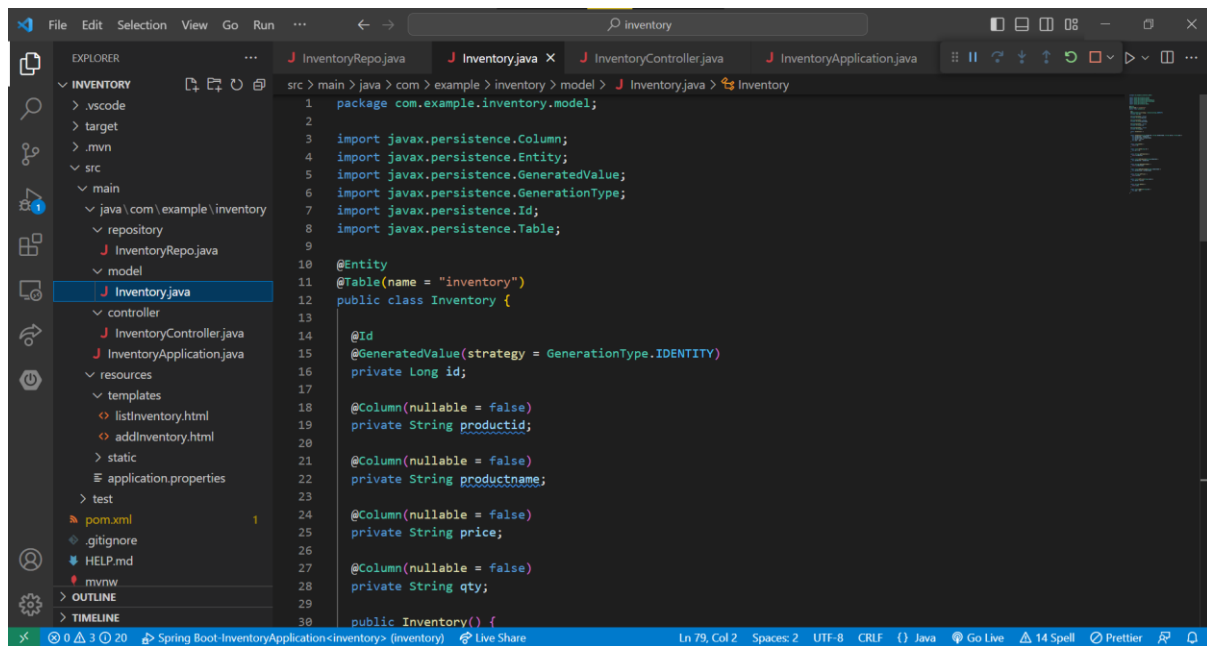
- Customizable handler mapping and view resolution. Handler mapping and view resolution strategies range from simple URL-based configuration, to sophisticated, purpose-built resolution strategies. Spring is more flexible than web MVC frameworks that mandate a particular technique.
- Flexible model transfer. Model transfer with a name/value Map supports easy integration with any view technology.
- Customizable locale and theme resolution, support for JSPs with or without Spring tag library, support for JSTL, support for Velocity without the need for extra bridges, and so on.
- A simple yet powerful JSP tag library known as the Spring tag library that provides support for features such as data binding and themes. The custom tags allow for maximum flexibility in terms of markup code.
- A JSP form tag library, introduced in Spring 2.0, that makes writing forms in JSP pages much easier.
- Beans whose lifecycle is scoped to the current HTTP request or HTTP Session. This is not a specific feature of Spring MVC itself, but rather of the `WebApplicationContext` container(s) that Spring MVC uses.

2. Problem Definition:

An MVC application for an inventory/list management system that allows the creation/addition and retrieval of items to the database.

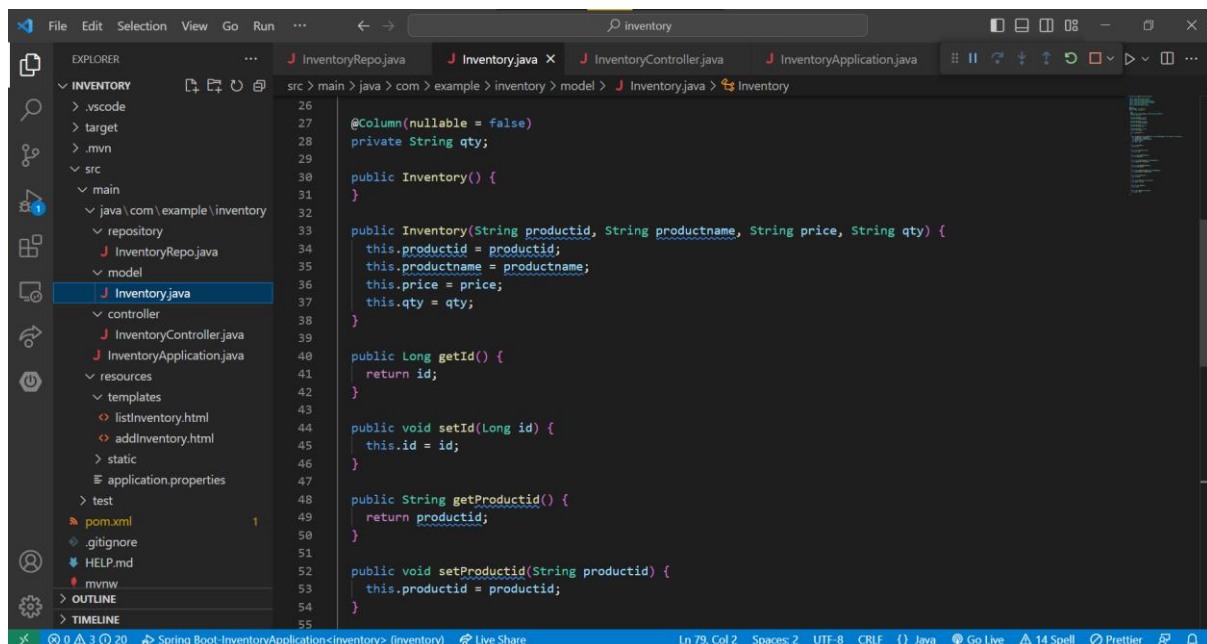
3. Code Screenshots:

Model:



```
1 package com.example.inventory.model;
2
3 import javax.persistence.Column;
4 import javax.persistence.Entity;
5 import javax.persistence.GeneratedValue;
6 import javax.persistence.GenerationType;
7 import javax.persistence.Id;
8 import javax.persistence.Table;
9
10 @Entity
11 @Table(name = "inventory")
12 public class Inventory {
13
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private Long id;
17
18     @Column(nullable = false)
19     private String productid;
20
21     @Column(nullable = false)
22     private String productname;
23
24     @Column(nullable = false)
25     private String price;
26
27     @Column(nullable = false)
28     private String qty;
29
30     public Inventory() {

```



```
26
27     @Column(nullable = false)
28     private String qty;
29
30     public Inventory() {
31     }
32
33     public Inventory(String productid, String productname, String price, String qty) {
34         this.productid = productid;
35         this.productname = productname;
36         this.price = price;
37         this.qty = qty;
38     }
39
40     public Long getId() {
41         return id;
42     }
43
44     public void setId(Long id) {
45         this.id = id;
46     }
47
48     public String getProductid() {
49         return productid;
50     }
51
52     public void setProductid(String productid) {
53         this.productid = productid;
54     }
55

```

The screenshot shows the VS Code editor with the `Inventory.java` file open in the `model` package. The file contains the following code:

```
53     this.productid = productid;
54 }
55
56 public String getProductname() {
57     return productname;
58 }
59
60 public void setProductname(String productname) {
61     this.productname = productname;
62 }
63
64 public String getPrice() {
65     return price;
66 }
67
68 public void setPrice(String price) {
69     this.price = price;
70 }
71
72 public String getQty() {
73     return qty;
74 }
75
76 public void setQty(String qty) {
77     this.qty = qty;
78 }
79 }
```

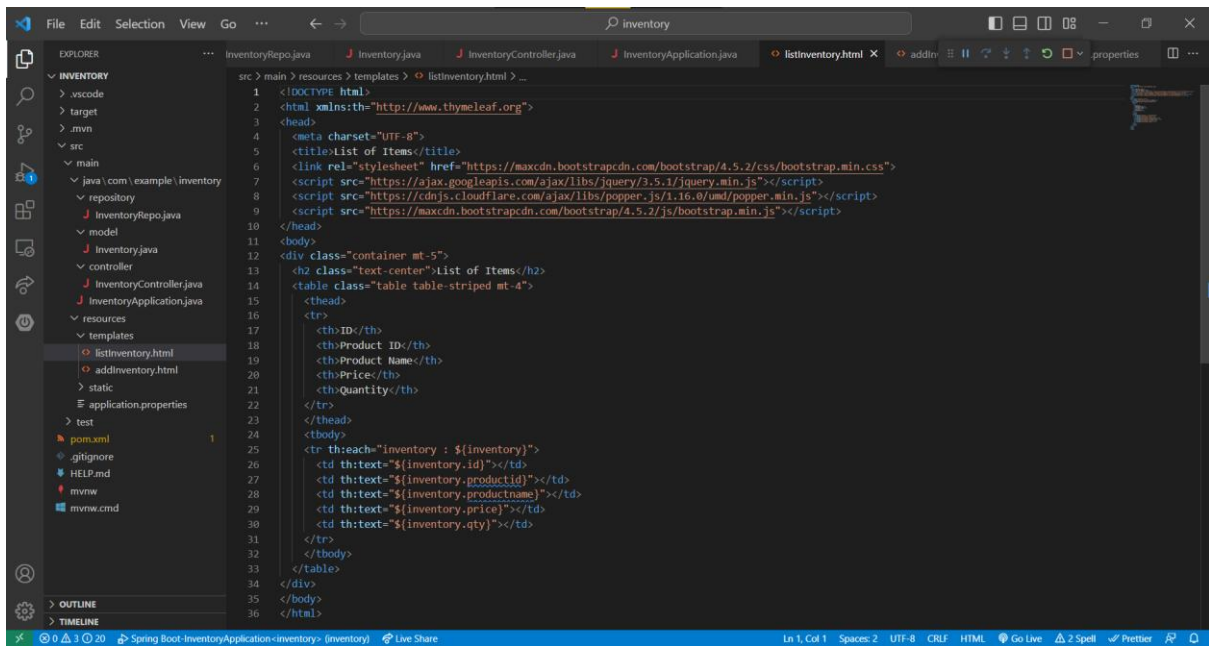
The Explorer sidebar on the left shows the project structure, with `Inventory.java` selected under the `model` package. The status bar at the bottom indicates the file is at line 79, column 2, in UTF-8 encoding with 2 spaces.

View:

The screenshot shows the VS Code editor with the `addInventory.html` file open in the `templates` package. The file contains the following HTML code:

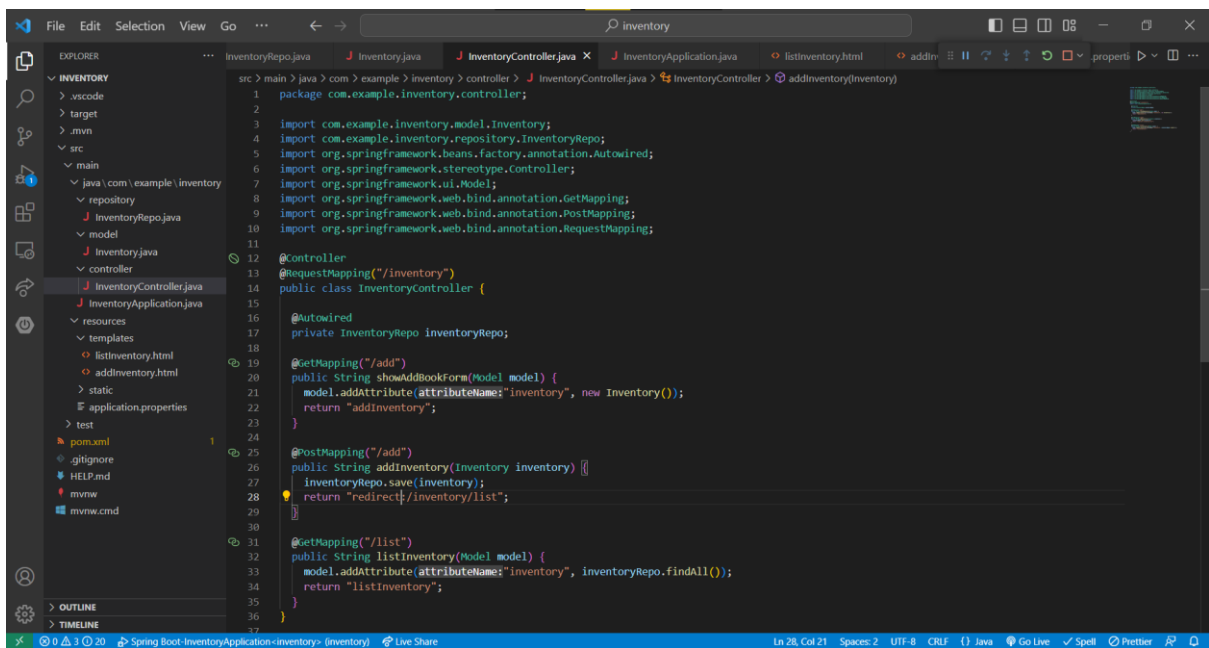
```
1 <!-->
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <meta charset="UTF-8">
5 <title>Add Item</title>
6 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
7 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
8 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
9 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
10 </head>
11 <body>
12 <div class="container mt-5">
13 <h2 class="text-center">Add New Item</h2>
14 <form th:action="@{/inventory/add}" th:object="${inventory}" method="post" class="mt-4">
15 <div class="form-group">
16 <label for="pid">Product ID:</label>
17 <input type="text" class="form-control" id="productid" placeholder="Enter Product ID" th:field="${productid}" required>
18 </div>
19 <div class="form-group">
20 <label for="pname">Product Name:</label>
21 <input type="text" class="form-control" id="productname" placeholder="Enter Product Name" th:field="${productname}" required>
22 </div>
23 <div class="form-group">
24 <label for="price">Price:</label>
25 <input type="text" class="form-control" id="price" placeholder="Enter Product Price" th:field="${price}" required>
26 </div>
27 <div class="form-group">
28 <label for="qty">Quantity:</label>
29 <input type="text" class="form-control" id="qty" placeholder="Enter Product Quantity" th:field="${qty}" required>
30 </div>
31 <button type="submit" class="btn btn-primary">Submit</button>
32 </form>
33 </div>
34 </body>
35 </html>
```

The Explorer sidebar on the left shows the project structure, with `addInventory.html` selected under the `templates` package. The status bar at the bottom indicates the file is at line 1, column 1, in UTF-8 encoding with 2 spaces.



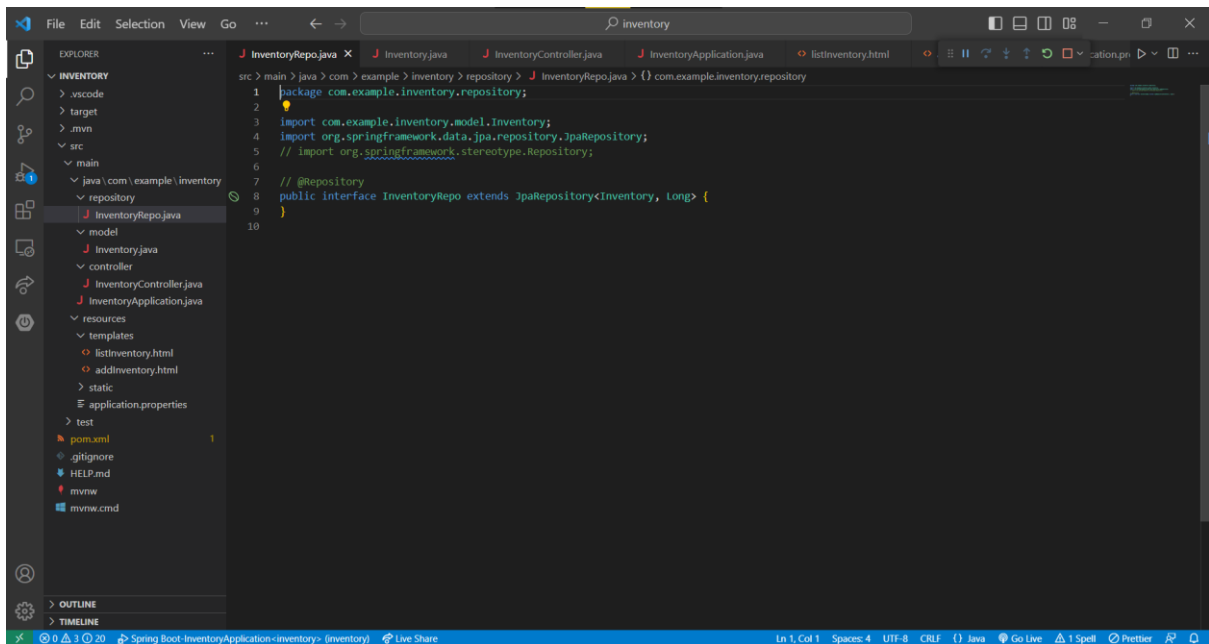
```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <meta charset="UTF-8">
5 <title>List of Items</title>
6 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
7 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
8 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
9 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
10 </head>
11 <body>
12 <div class="container mt-5">
13 <h2 class="text-center">List of Items</h2>
14 <table class="table table-striped mt-4">
15 <thead>
16 <tr>
17 <th>ID</th>
18 <th>Product ID</th>
19 <th>Product Name</th>
20 <th>Price</th>
21 <th>Quantity</th>
22 </tr>
23 </thead>
24 <tbody>
25 <tr th:each="inventory : ${inventory}">
26 <td th:text="${inventory.id}"></td>
27 <td th:text="${inventory.productId}"></td>
28 <td th:text="${inventory.productName}"></td>
29 <td th:text="${inventory.price}"></td>
30 <td th:text="${inventory.qty}"></td>
31 </tr>
32 </tbody>
33 </table>
34 </div>
35 </body>
36 </html>
```

Controller:

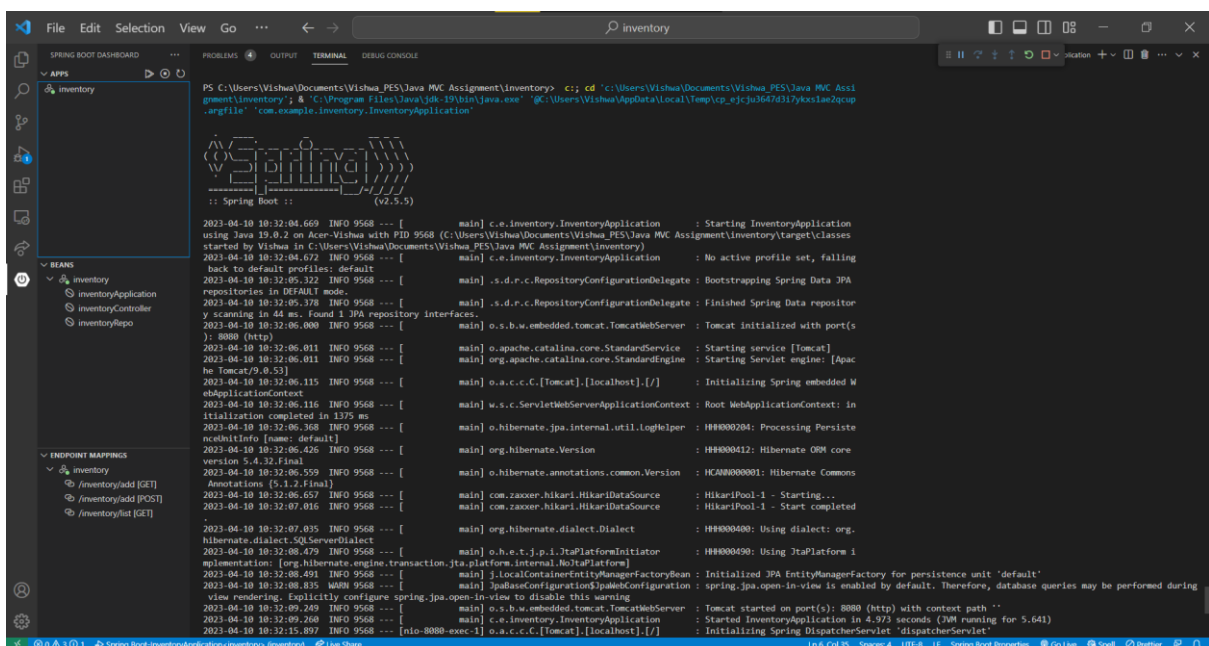


```
1 package com.example.inventory.controller;
2
3 import com.example.inventory.model.Inventory;
4 import com.example.inventory.repository.InventoryRepo;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.ui.Model;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.PostMapping;
10 import org.springframework.web.bind.annotation.RequestMapping;
11
12 @Controller
13 @RequestMapping("/inventory")
14 public class InventoryController {
15
16     @Autowired
17     private InventoryRepo inventoryRepo;
18
19     @GetMapping("/add")
20     public String showAddBookForm(Model model) {
21         model.addAttribute("inventory", new Inventory());
22         return "addInventory";
23     }
24
25     @PostMapping("/add")
26     public String addInventory(Inventory inventory) {
27         inventoryRepo.save(inventory);
28         return "redirect:/inventory/list";
29     }
30
31     @GetMapping("/list")
32     public String listInventory(Model model) {
33         model.addAttribute("inventory", inventoryRepo.findAll());
34         return "listInventory";
35     }
36 }
37
```

Database/Repository:



4. Console Screenshot:



5. UI Screenshots:

Browser: Add Item | localhost:8080/inventory/add

Add New Item

Product ID:

Product Name:

Price:

Quantity:

Browser: List of Items | localhost:8080/inventory/list

List of Items

ID	Product ID	Product Name	Price	Quantity
3	123456789	Milk Bread	50	2
4	123456785	Milk	18	1
5	123656785	Noodles	20	1

6. Database Screenshots:

SQLQuery3.sql - ACER-VISHWA\SQLEXPRESS01.inventorydb (vishwa2 (70)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

- Connect
- ACER-VISHWA\SQLEXPRESS01 (S)
- Databases
 - System Databases
 - Database Snapshots
 - bookstoredb
 - inventory_management_sys
 - inventory_management_system
 - inventorydb
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.inventory
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store
 - Service Broker
 - Storage
 - Security
 - Security
 - Logins

SQLQuery3.sql - AC...rydb (vishwa2 (70))

```
CREATE TABLE inventory (  
    id bigint IDENTITY(1,1) NOT NULL,  
    productid varchar(50) NOT NULL,  
    productname varchar(50) NOT NULL,  
    price varchar(50) NOT NULL,  
    qty varchar(50) NOT NULL  
)
```

100 %

Messages

Commands completed successfully.

Completion time: 2023-04-07T19:04:47.4332765+05:30

100 %

Query executed successfully. ACER-VISHWA\SQLEXPRESS01 (1... vishwa2 (70) inventorydb 00:00:00 0 rows

Ready Ln 7 Col 2 Ch 2 INS

SQLQuery2.sql - ACER-VISHWA\SQLEXPRESS01.inventorydb (vishwa2 (52)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

- Connect
- ACER-VISHWA\SQLEXPRESS01 (S)
- Databases
 - System Databases
 - Database Snapshots
 - bookstoredb
 - inventory_management_sys
 - inventory_management_system
 - inventorydb
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.inventory
 - Views
 - External Resources
 - Synonyms
 - Programmability
 - Query Store
 - Service Broker
 - Storage
 - Security
 - Security
 - Server Objects

SQLQuery2.sql - AC...rydb (vishwa2 (52))

```
select * from inventory;
```

100 %

Results

	id	price	productid	productname	qty
1	3	50	123456789	Milk Bread	2
2	4	18	123456785	Milk	1
3	5	20	123656785	Noodles	1

Messages

Query executed successfully. ACER-VISHWA\SQLEXPRESS01 (1... vishwa2 (52) inventorydb 00:00:00 3 rows

Ready Ln 1 Col 25 Ch 25 INS