Application.properties :

```
server.port = 2215

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/practise_during_creation
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#server.servlet.context-path=/myapp

spring.jpa.show-sql: true
spring.jpa.open-in-view=true

#debug=true

#spring.profiles.active=dev

#root@localhost:3306

#logging.level.org.springframework.data.jpa=DEBUG
#logging.level.org.hibernate.SQL=DEBUG
#logging.level.org.springframework.security=DEBUG
#logging.level.org.springframework.web: DEBUG
```

POM.XML :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <parent>
            <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.1</version>
            <relativePath/> <!-- lookup parent from repository -->
      </parent>
      <groupId>com.cdac_project</groupId>
      <artifactId>DrugDesributionManagementSystem</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>DrugDesributionManagementSystem</name>
```

```
      <description>CDAC Final project named as
&amp;quot;DrugDesributionManagementSystem&amp;quot; based on Spring Boot ,
MySQL and ReactJS , i.e. ,Full Stack Project</description>
      <properties>
            <java.version>17</java.version>
    <spring.version>5.3.14</spring.version>
    <spring-boot.version>2.6.1</spring-boot.version>
    <spring-boot-devtools.version>2.6.1</spring-boot-
devtools.version>
            <maven.compiler.source>17</maven.compiler.source>
            <maven.compiler.target>17</maven.compiler.target>
    </properties>
<dependencies>
<!--
      <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
-->

<dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId>jaxb-runtime</artifactId>
    <version>2.3.5</version>
    <exclusions>
        <exclusion>
            <groupId>jakarta.xml.bind</groupId>
            <artifactId>jakarta.xml.bind-api</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>txw2</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>istack-commons-runtime</artifactId>
        </exclusion>
        <exclusion>
            <groupId>stax</groupId>
            <artifactId>stax-api</artifactId>
        </exclusion>
        <exclusion>
            <groupId>stax</groupId>
            <artifactId>stax-ex</artifactId>
        </exclusion>
        <exclusion>
            <groupId>com.sun.xml.fastinfoset</groupId>
            <artifactId>FastInfoset</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.glassfish</groupId>
            <artifactId>javax.activation</artifactId>
        </exclusion>
    </exclusions>
</dependency>

<!-- -->
```

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
</dependency>

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId> <!-- Or the version
you are using -->
</dependency>


<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
</dependency>

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

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>


<!--
https://mvnrepository.com/artifact/org.springframework.boot/spring-
boot-devtools -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional>
</dependency>

<dependency>
    <groupId>com.mchange</groupId>
    <artifactId>mchange-commons-java</artifactId>
    <version>0.2.11</version>
</dependency>

    <!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-
runtime -->
```

```
        <dependency>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>jaxb-runtime</artifactId> <!-- Update to the
correct version if needed -->
        </dependency>

        <dependency>
            <groupId>jakarta.xml.bind</groupId>
            <artifactId>jakarta.xml.bind-api</artifactId>
        </dependency>

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>


<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
</dependency>

 <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
</dependency>



<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
</dependency>
</dependencies>

<!-- Named Query 21-02-2024 -->



<!-- -->
      <build>
            <plugins>
```

```
                <plugin>
                      <groupId>org.springframework.boot</groupId>
                      <artifactId>spring-boot-maven-
plugin</artifactId>
                      <configuration>
                            <image>

      <builder>paketobuildpacks/builder-jammy-base:latest</builder>
<!-- Verify compatibility with Java 1.8 -->
                            </image>
                      </configuration>
                </plugin>
                 <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>17</source>
                <target>17</target>
            </configuration>
        </plugin>
            </plugins>
      </build>

------------------------------------------------------------------------
----

</project>
Main Function Project :
------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.security.servlet.SecurityAutoC
onfiguration;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan("com.cdac_project.*")
public class DrugDesributionManagementSystemApplication {

      public static void main(String[] args) {
            System.out.println("Running");

      SpringApplication.run(DrugDesributionManagementSystemApplicatio
n.class, args);
      }

}

MODELS :-

Address :
  ~>
```

```
   ------------------------------------------------------------------------
---------------------------------------------- package
com.cdac_project.model;

import java.util.*;

import javax.persistence.*;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name = "address")
//@NamedQuery(name = "Address.findByPharmacistid",
//query = "SELECT a FROM Address a WHERE a.pharmacist_id =
:pharmacistId")
public class Address {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "address_id")
    private int addressid;

    @Column(name = "pharmacist_id")
    private int pharmacistid;

    @Column(name = "pharmacist_name")
    private String name;

    @Column(name = "Full_Address")
    private String FullAddress;

      public Address() {
            super();
      }

      public Address(int addressid, int pharmacistid, String name,
String fullAddress) {
            super();
            this.addressid = addressid;
            this.pharmacistid = pharmacistid;
            this.name = name;
            FullAddress = fullAddress;
      }

      public Address(int pharmacistid, String name, String
fullAddress) {
            super();
            this.pharmacistid = pharmacistid;
            this.name = name;
            FullAddress = fullAddress;
      }

      public int getAddressid() {
            return addressid;
      }

      public void setAddressid(int addressid) {
```

```
		this.addressid = addressid;
	}

	public int getPharmacistid() {
		return pharmacistid;
	}

	public void setPharmacistid(int pharmacistid) {
		this.pharmacistid = pharmacistid;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getFullAddress() {
		return FullAddress;
	}

	public void setFullAddress(String fullAddress) {
		FullAddress = fullAddress;
	}
}
Bill :
	~>
	 ------------------------------------------------------------------
---------------------------------------------------- package
com.cdac_project.model;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.*;

import javax.persistence.*;

import com.fasterxml.jackson.annotation.JsonIgnore;


@Entity
@Table(name =  "bill_table")
public class Bill {

	@Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "bill_id")
    private int id;

    @Column(name = "Order_id")
    private int orderId;

    @Column(name = "pharmacist_id")
    private int pharmacistId;
```

```
    @Column(name = "Pharmacist_Name")
    private String pharmacistName;

    @Column(name = "medicine_id")
    private int medicineId;

    @Column(name = "Medicine_name")
    @Transient
    private String medicineName;

    @Column(name = "Quantity")
    private int quantity;

    @Column(name = "Total_Amount")
    private double totalAmount;

    @Column(name = "Discount_price")
    private int discountPrice;

    @Column(name = "Discounted_price")
    private int discountedPrice;

    @Column(name = "To_Pay_Amount")
    private int toPayAmount;

    @Column(name = "Billing_Date")
    private LocalDateTime billingDate;

      public Bill() {
            super();
      }

      public Bill(int id, int orderId, int pharmacistId, String
pharmacistName, int medicineId,
                  int quantity, double totalAmount, int
discountPrice, int discountedPrice, int toPayAmount,
                  LocalDateTime billingDate) {
            super();
            this.id = id;
            this.orderId = orderId;
            this.pharmacistId = pharmacistId;
            this.pharmacistName = pharmacistName;
            this.medicineId = medicineId;
            //this.medicineName = medicineName;
            this.quantity = quantity;
            this.totalAmount = totalAmount;
            this.discountPrice = discountPrice;
            this.discountedPrice = discountedPrice;
            this.toPayAmount = toPayAmount;
            this.billingDate = billingDate;
      }

      public int getId() {
            return id;
      }

      public void setId(int id) {
```

```
		this.id = id;
	}

	public int getOrderId() {
		return orderId;
	}

	public void setOrderId(int orderId) {
		this.orderId = orderId;
	}

	public int getPharmacistId() {
		return pharmacistId;
	}

	public void setPharmacistId(int pharmacistId) {
		this.pharmacistId = pharmacistId;
	}

	public String getPharmacistName() {
		return pharmacistName;
	}

	public void setPharmacistName(String pharmacistName) {
		this.pharmacistName = pharmacistName;
	}

	public int getMedicineId() {
		return medicineId;
	}

	public void setMedicineId(int medicineId) {
		this.medicineId = medicineId;
	}

//	public String getMedicineName() {
//		return medicineName;
//	}
//
//	public void setMedicineName(String medicineName) {
//		this.medicineName = medicineName;
//	}

	public int getQuantity() {
		return quantity;
	}

	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}

	public double getTotalAmount() {
		return totalAmount;
	}

	public void setTotalAmount(double totalAmount) {
		this.totalAmount = totalAmount;
```

```
	}

	public int getDiscountPrice() {
		return discountPrice;
	}

	public void setDiscountPrice(int discountPrice) {
		this.discountPrice = discountPrice;
	}

	public int getDiscountedPrice() {
		return discountedPrice;
	}

	public void setDiscountedPrice(int discountedPrice) {
		this.discountedPrice = discountedPrice;
	}

	public int getToPayAmount() {
		return toPayAmount;
	}

	public void setToPayAmount(int toPayAmount) {
		this.toPayAmount = toPayAmount;
	}

	public LocalDateTime getBillingDate() {
		return billingDate;
	}

	public void setBillingDate(LocalDateTime billingDate) {
		this.billingDate = billingDate;
	}


}
Cart :
~>-------------------------------------------------------------------
---------------------------------------------- package
com.cdac_project.model;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.*;

@Entity
@Table(name = "cart", uniqueConstraints =
@UniqueConstraint(columnNames = {"pharmacist_id"}))
public class Cart {
	@Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "cart_id")
    private int id;

	@OneToOne(fetch = FetchType.LAZY)
	@JoinColumn(name = "pharmacist_id", nullable = false)
```

```
	private Pharmacist pharmacist;

	@OneToMany(mappedBy = "cart" , cascade = CascadeType.ALL,
orphanRemoval = true)
	@Column(name = "cart_medicines")
	private Set<CartMedicine> cartMedicine = new HashSet<>();

	@Column(name="total_item")
	private int totalItem;

	private int totalPrice;

	public Cart(int id, Pharmacist pharmacist, Set<CartMedicine>
cartMedicine, int totalItem, int totalPrice) {
		super();
		this.id = id;
		this.pharmacist = pharmacist;
		this.cartMedicine = cartMedicine;
		this.totalItem = totalItem;
		this.totalPrice = totalPrice;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public Pharmacist getPharmacist() {
		return pharmacist;
	}

	public void setPharmacist(Pharmacist pharmacist) {
		this.pharmacist = pharmacist;
	}

	public Set<CartMedicine> getCartMedicine() {
		return cartMedicine;
	}

	public void setCartMedicine(Set<CartMedicine> cartMedicine) {
		this.cartMedicine = cartMedicine;
	}

	public int getTotalItem() {
		return totalItem;
	}

	public void setTotalItem(int totalItem) {
		this.totalItem = totalItem;
	}

	public int getTotalPrice() {
		return totalPrice;
	}
```

```
	public void setTotalPrice(int totalPrice) {
		this.totalPrice = totalPrice;
	}

	public Cart() {
		super();
	}


}

CartMedicine :
		~>
		 ------------------------------------------------------------
---------------------------------------------------- package
com.cdac_project.model;

import javax.persistence.*;

import org.hibernate.annotations.Immutable;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity(name="cart_medicine")
public class CartMedicine {


	  @Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "cart_id")
	private int id;

	@Column(name = "pharmacist_id")
	private int pharmacistId;

	@Column(name = "medicine_id")
	private int medicineId;

   // @JsonIgnore
	@ManyToOne
	private Cart cart;


	@ManyToOne
	private Medicine medicine;

	@Column(name = "quantity")
	private int quantity;

	private  int price ;

	  public Cart getCart() {
		return cart;
	  }

	  public void setCart(Cart cart) {
```

```
		this.cart = cart;
	}

	public Medicine getMedicine() {
		return medicine;
	}

	public void setMedicine(Medicine medicine) {
		this.medicine = medicine;
	}

	public int getPrice() {
		return price;
	}

	public void setPrice(int price) {
		this.price = price;
	}

	public CartMedicine() {
		super();
	}

	public CartMedicine(int id, int pharmacistId, int medicineId,
int quantity) {
		super();
		this.id = id;
		this.pharmacistId = pharmacistId;
		this.medicineId = medicineId;
		this.quantity = quantity;
	}



	public CartMedicine(int id, int pharmacistId, Cart cart, int
medicineId, Medicine medicine, int quantity,
			int price) {
		super();
		this.id = id;
		this.pharmacistId = pharmacistId;
		this.cart = cart;
		this.medicineId = medicineId;
		this.medicine = medicine;
		this.quantity = quantity;
		this.price = price;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public int getPharmacistId() {
		return pharmacistId;
```

```
	}

	public void setPharmacistId(int pharmacistId) {
		this.pharmacistId = pharmacistId;
	}

	public int getMedicineId() {
		return medicineId;
	}

	public void setMedicineId(int medicineId) {
		this.medicineId = medicineId;
	}

	public int getQuantity() {
		return quantity;
	}

	public void setQuantity(int quantity) {
		this.quantity = quantity;
	}



}
Distributor :
        ~>
         ------------------------------------------------------------
---------------------------------------------------- package
com.cdac_project.model;

import javax.persistence.*;

@Entity
@Table(name = "distributor_db", uniqueConstraints =
@UniqueConstraint(columnNames = "distributorEmail"))
public class Distributor {


	@Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Distributor_ID")
    private int id;

    @Column(name = "distributor_name")
    private String name;

    @Column(name = "distributorEmail")
    private String email;

    @Column(name = "Password")
    private String password;

	public Distributor() {
		super();
	}
```

```
	public Distributor(int id, String name, String email, String
password) {
		super();
		this.id = id;
		this.name = name;
		this.email = email;
		this.password = password;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword String password) {
		this.password = password;
	}



}
```

Medicine :

```
		~>
		 ------------------------------------------------------------
------------------------------------------------------------ package
com.cdac_project.model;

import java.time.LocalDate;
import java.util.*;

import javax.persistence.*;

@Entity
```

```
@Table(name = "medicine_db", uniqueConstraints =
@UniqueConstraint(columnNames = "Medicine_Name"))
public class Medicine {


      @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Medicine_id")
    private int id;

    @Column(name = "Medicine_name")
    private String name;

    @ManyToOne
    @JoinColumn(name = "Category_id")
    private MedicineCategory categoryId;

    @Column(name = "Medicine_Quantity")
    private int quantity;

    @Column(name = "Manufacture_date")
    private LocalDate manufactureDate;

    @Column(name = "Unit_Price")
    private int unitPrice;

    @ManyToOne
    @Transient
    private Order order;

      public Medicine() {
            super();
      }



      public Medicine(int id, String name, MedicineCategory
categoryId, int quantity, LocalDate manufactureDate,
                  int unitPrice) {
            super();
            this.id = id;
            this.name = name;
            this.categoryId = categoryId;
            this.quantity = quantity;
            this.manufactureDate = manufactureDate;
            this.unitPrice = unitPrice;
      }



      public int getId() {
            return id;
      }



      public void setId(int id) {
```

```
	this.id = id;
}



public String getName() {
	return name;
}



public void setName(String name) {
	this.name = name;
}



public MedicineCategory getCategoryId() {
	return categoryId;
}



public void setCategoryId(MedicineCategory categoryId) {
	this.categoryId = categoryId;
}



public int getQuantity() {
	return quantity;
}



public void setQuantity(int quantity) {
	this.quantity = quantity;
}



public LocalDate getManufactureDate() {
	return manufactureDate;
}



public void setManufactureDate(LocalDate manufactureDate) {
	this.manufactureDate = manufactureDate;
}



public int getUnitPrice() {
	return unitPrice;
}
```

```
	public void setUnitPrice(int unitPrice) {
		this.unitPrice = unitPrice;
	}



	public Order getOrder() {
		return order;
	}



	public void setOrder(Order order) {
		this.order = order;
	}




}
MedicineCategory :
           ~>
            ------------------------------------------------------------
---------------------------------------------------------- package
com.cdac_project.model;

import java.time.LocalDate;
import java.util.*;

import javax.persistence.*;

@Entity
@Table(name = "medicine_db", uniqueConstraints =
@UniqueConstraint(columnNames = "Medicine_Name"))
public class Medicine {


	  @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Medicine_id")
    private int id;

    @Column(name = "Medicine_name")
    private String name;

    @ManyToOne
    @JoinColumn(name = "Category_id")
    private MedicineCategory categoryId;

    @Column(name = "Medicine_Quantity")
    private int quantity;

    @Column(name = "Manufacture_date")
    private LocalDate manufactureDate;
```

```
    @Column(name = "Unit_Price")
    private int unitPrice;

    @ManyToOne
    @Transient
    private Order order;

      public Medicine() {
            super();
      }



      public Medicine(int id, String name, MedicineCategory
categoryId, int quantity, LocalDate manufactureDate,
                  int unitPrice) {
            super();
            this.id = id;
            this.name = name;
            this.categoryId = categoryId;
            this.quantity = quantity;
            this.manufactureDate = manufactureDate;
            this.unitPrice = unitPrice;
      }



      public int getId() {
            return id;
      }



      public void setId(int id) {
            this.id = id;
      }



      public String getName() {
            return name;
      }



      public void setName(String name) {
            this.name = name;
      }



      public MedicineCategory getCategoryId() {
            return categoryId;
      }
```

```
public void setCategoryId(MedicineCategory categoryId) {
	this.categoryId = categoryId;
}



public int getQuantity() {
	return quantity;
}



public void setQuantity(int quantity) {
	this.quantity = quantity;
}



public LocalDate getManufactureDate() {
	return manufactureDate;
}



public void setManufactureDate(LocalDate manufactureDate) {
	this.manufactureDate = manufactureDate;
}



public int getUnitPrice() {
	return unitPrice;
}



public void setUnitPrice(int unitPrice) {
	this.unitPrice = unitPrice;
}



public Order getOrder() {
	return order;
}



public void setOrder(Order order) {
	this.order = order;
}
```

```
}
Order :
          ~>
          ------------------------------------------------------------
---------------------------------------------------- package
com.cdac_project.model;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.*;

import javax.persistence.*;

@Entity
@Table(name = "order_db")
public class Order {

      @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "order_id")
    private int id;

    @OneToMany(mappedBy ="order")
    private List<OrderMedicine> om = new ArrayList<>();

    @Column(name = "Medicine_id")
    private int medicineId;

    @Column(name = "Medicine_Name")
    @Transient
    private String medicineName;

    @Column(name = "pharmacist_id")
    private int pharmacistId;

    @Column(name = "pharmacist_name")
    @Transient
    private String pharmacistName;

    @ManyToOne
    private Pharmacist pharmacist;

    @OneToOne
    @Transient
    private Address ShippingAddress;

    @Column(name = "ShippingAddress_id")
    //@Transient
    private int ShippingAddress_id;

      @Column(name = "Price")
    private double price;

    @Column(name = "Order_Date")
    private LocalDateTime orderDate;

    @Column(name = "Delivery_Date")
```

```
    private LocalDateTime deliveryDate;

    @Column(name = "bill_id")
    private int billId;

    @Column(name = "createdAt")
    private LocalDateTime createdAt;

    @Transient
    public OrderStatus status;



      public OrderStatus getStatus() {
            return status;
      }

      public void setStatus(OrderStatus delivered) {
            this.status = delivered;
      }

      public Order(int id, List<OrderMedicine> om, int medicineId,
String medicineName, int pharmacistId,
                  String pharmacistName, Pharmacist pharmacist,
Address shippingAddress, int shippingAddress_id, double price,
                  LocalDateTime orderDate, LocalDateTime
deliveryDate, int billId, LocalDateTime createdAt, OrderStatus
status) {
            super();

            this.id = id;
            this.om = om;
            this.medicineId = medicineId;
            this.medicineName = medicineName;
            this.pharmacistId = pharmacistId;
            this.pharmacistName = pharmacistName;
            this.pharmacist = pharmacist;
            ShippingAddress = shippingAddress;
            ShippingAddress_id = shippingAddress_id;
            this.price = price;
            this.orderDate = orderDate;
            this.deliveryDate = deliveryDate;
            this.billId = billId;
            this.createdAt = createdAt;
            this.status = status;
      }

      public int getId() {
            return id;
      }

      public void setId(int id) {
            this.id = id;
      }
```

```
public int getShippingAddress_id() {
        return ShippingAddress_id;
  }

  public void setShippingAddress_id(int shippingAddress_id) {
        ShippingAddress_id = shippingAddress_id;
  }



  public List<OrderMedicine> getOm() {
        return om;
  }

  public void setOm(List<OrderMedicine> om) {
        this.om = om;
  }

  public int getMedicineId() {
        return medicineId;
  }

  public void setMedicineId(int medicineId) {
        this.medicineId = medicineId;
  }

  public Pharmacist getPharmacist() {
        return pharmacist;
  }

  public void setPharmacist(Pharmacist pharmacist) {
        this.pharmacist = pharmacist;
  }

  public Address getShippingAddress() {
        return ShippingAddress;
  }

  public void setShippingAddress(Address shippingAddress) {
        ShippingAddress = shippingAddress;
  }

  public String getMedicineName() {
        return medicineName;
  }

  public void setMedicineName(String medicineName) {
        this.medicineName = medicineName;
  }

  public int getPharmacistId() {
        return pharmacistId;
  }

  public void setPharmacistId(int pharmacistId) {
        this.pharmacistId = pharmacistId;
  }
```

```
	public String getPharmacistName() {
		return pharmacistName;
	}

	public void setPharmacistName(String pharmacistName) {
		this.pharmacistName = pharmacistName;
	}

	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public LocalDateTime getOrderDate() {
		return orderDate;
	}

	public void setOrderDate(LocalDateTime orderDate) {
		this.orderDate = orderDate;
	}

	public LocalDateTime getDeliveryDate() {
		return deliveryDate;
	}

	public void setDeliveryDate(LocalDateTime deliveryDate) {
		this.deliveryDate = deliveryDate;
	}

	public int getBillId() {
		return billId;
	}

	public void setBillId(int billId) {
		this.billId = billId;
	}

	public LocalDateTime getCreatedAt() {
		return createdAt;
	}

	public void setCreatedAt(LocalDateTime createdAt) {
		this.createdAt = createdAt;
	}
```

```
		public Order(List<OrderMedicine> om, int medicineId,
String medicineName, int pharmacistId, String pharmacistName,
```

```
                  Pharmacist pharmacist, Address shippingAddress, int
shippingAddress_id, double price,
                  LocalDateTime orderDate, LocalDateTime
deliveryDate, int billId, LocalDateTime createdAt) {
            super();
            this.om = om;
            this.medicineId = medicineId;
            this.medicineName = medicineName;
            this.pharmacistId = pharmacistId;
            this.pharmacistName = pharmacistName;
            this.pharmacist = pharmacist;
            ShippingAddress = shippingAddress;
            ShippingAddress_id = shippingAddress_id;
            this.price = price;
            this.orderDate = orderDate;
            this.deliveryDate = deliveryDate;
            this.billId = billId;
            this.createdAt = createdAt;
      }

            public Order(int id, List<OrderMedicine> om, int
medicineId, String medicineName, int pharmacistId,
                  String pharmacistName, Pharmacist pharmacist,
Address shippingAddress, int shippingAddress_id, double price,
                  LocalDateTime orderDate, LocalDateTime
deliveryDate, int billId, LocalDateTime createdAt) {
            super();
            this.id = id;
            this.om = om;
            this.medicineId = medicineId;
            this.medicineName = medicineName;
            this.pharmacistId = pharmacistId;
            this.pharmacistName = pharmacistName;
            this.pharmacist = pharmacist;
            ShippingAddress = shippingAddress;
            ShippingAddress_id = shippingAddress_id;
            this.price = price;
            this.orderDate = orderDate;
            this.deliveryDate = deliveryDate;
            this.billId = billId;
            this.createdAt = createdAt;
      }

            public Order() {
            super();
      }


}
OrderStatus :
~>
-------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.model;

public enum OrderStatus {
    PLACED,
```

```
    CONFIRMED,
    SHIPPED,
    DELIVERED,
    CANCELED


}

Pharmacist :
              ~>
               ------------------------------------------------------
------------------------------------------------------------ package
com.cdac_project.model;

import java.util.*;

import javax.persistence.*;


@Entity
@Table(name = "pharmacist_db", uniqueConstraints =
@UniqueConstraint(columnNames = "pharmacist_Email"))
public class Pharmacist {
          @Id
          @GeneratedValue(strategy = GenerationType.IDENTITY)
          @Column(name = "pharmacist_id")
          private Integer id;

          @Column(name = "pharmacist_name")
          private String name;

          @Column(name = "License_Number")
          private String licenseNumber;

          @Column(name = "pharmacist_Email")
          private String email;

          @Column(name = "Address")
          private String address;

          @Column(name = "Password")
          private String password;

          @OneToMany(mappedBy = "pharmacist")
          private List<Order> orders;

            public Pharmacist(int id, String name, String
licenseNumber, String email, String address, String password) {
                  super();
                  this.id = id;
                  this.name = name;
                  this.licenseNumber = licenseNumber;
                  this.email = email;
                  this.address = address;
                  this.password = password;
            }
```

```
	public Integer getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getLicenseNumber() {
		return licenseNumber;
	}

	public void setLicenseNumber(String licenseNumber) {
		this.licenseNumber = licenseNumber;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public String getAddress() {
		return address;
	}

	public void setAddress(String addressId) {
		this.address = addressId;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public Pharmacist() {
		super();
	}



}
```

TotaleRoles :

```
            ~>
            ------------------------------------------------------------
---------------------------------------------------------- package
com.cdac_project.model;

public enum TotalRoles {

      Distributer,
      Pharmacist
}

------------------------------------------------------------------------
----

Configuration : ~>

------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.config;

import org.springframework.boot.jdbc.DataSourceBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.*;

import java.sql.Connection;
import java.sql.DriverManager;

import javax.sql.DataSource;
import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.CorsRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class DataSourceConfig implements WebMvcConfigurer {


    @Bean
    public DataSource dataSource() {
        return DataSourceBuilder
                .create()

.url("jdbc:mysql://localhost:3306/practise_during_creation")
                .username("root")
                .password("root")
                .driverClassName("com.mysql.cj.jdbc.Driver")
                .build();
    }
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedOrigins("http://localhost:3000") // Add your
front-end URL here
                .allowedMethods("GET", "POST", "PUT", "DELETE")
                .allowedHeaders("*");
    }
```

```
}
-----------------------------------------------------------------------
----

Controller :~>

AddressController :
                -----------------------------------------------------
---------------------------------------------------------- package
com.cdac_project.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.cdac_project.exception.AddressNotFoundException;
import com.cdac_project.model.Address;
import com.cdac_project.service.AddressService;

import java.util.List;

@RestController
@RequestMapping("/address")
@CrossOrigin(origins = "http://localhost:3000")
public class AddressController {

      @Autowired
    private final AddressService addressService;

    public AddressController(AddressService addressService) {
        this.addressService = addressService;
    }

    @PostMapping("/add")
    public ResponseEntity<Address> addAddress(@RequestBody Address
address) {
      System.out.println("In Add Address Method");
        Address savedAddress = addressService.addAddress(address);
        return
ResponseEntity.status(HttpStatus.CREATED).body(savedAddress);
    }

    @GetMapping("/{addressId}")
    public ResponseEntity<Address> getAddressById(@PathVariable int
addressId) {
      System.out.println("In get Address Method");
        try {
            Address address =
addressService.getAddressById(addressId);
            return ResponseEntity.ok(address);
        } catch (AddressNotFoundException e) {
            return
ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }
```

```
    @GetMapping "/pharmacist/{pharmacistId}"
    public ResponseEntity<List<Address>>
getAddressesByPharmacistId @PathVariable int pharmacistId {
      System.out.println("In get Address Method by id");
        List<Address> addresses =
addressService.getAddressesByPharmacistid(pharmacistId);
        return ResponseEntity.ok(addresses);
    }

    @PutMapping "/update/{addressId}"
    public ResponseEntity<Address> updateAddress(@PathVariable int
addressId, @RequestBody Address address) {
      System.out.println("In Update Address Method");
        try {
            Address updatedAddress =
addressService.updateAddress(addressId, address);
            return ResponseEntity.ok(updatedAddress);
        } catch (AddressNotFoundException e) {
            return
ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }

    @DeleteMapping("/delete/{addressId}")
    public ResponseEntity<String> deleteAddress(@PathVariable int
addressId) {
      System.out.println("In delete Address Method");
        try {
            addressService.deleteAddress(addressId);
            return ResponseEntity.ok("Address deleted successfully");
        } catch (AddressNotFoundException e) {
            return
ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }
}


CartController :
            ------------------------------------------------------------
------------------------------------------------------------ package
com.cdac_project.controller;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.cdac_project.exception.CartException;
import com.cdac_project.exception.CartIsEmptyExcetpion;
import com.cdac_project.exception.CartMedicineException;
import com.cdac_project.exception.MedicineException;
import com.cdac_project.exception.PharmacistException;
import com.cdac_project.model.Cart;
import com.cdac_project.request.AddMedicineRequest;
import com.cdac_project.service.CartService;

@RestController
@RequestMapping("/cart")
```

```
@CrossOrigin(origins = "http://localhost:3000")
public class CartController {

    private CartService cartService;

    public CartController(CartService cartService) {
        this.cartService = cartService;
    }

    @PostMapping("/create/{pharmacistId}")
    public ResponseEntity<String> createCart(@PathVariable int
pharmacistId) throws PharmacistException {
      System.out.println("In Add of Create Cart Method");
        Cart createdCart = cartService.createCart(pharmacistId);
        return ResponseEntity.ok("Cart created successfully with ID:
" + createdCart.getId());
    }

    @PostMapping("/addMedicine")
    public ResponseEntity<String> addMedicineToCart(@RequestBody
AddMedicineRequest request) {
      System.out.println("In Adding medicine of Create Cart Method");
        try {
            String message =
cartService.addCartMedicine(request.getpID(), request.getQuantity());
            return ResponseEntity.ok(message);
        } catch (MedicineException | CartException |
CartMedicineException | PharmacistException e) {
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
        }
    }

    @GetMapping("/get/{pharmacistId}")
    public ResponseEntity<Cart> getCartByPharmacistId(@PathVariable
int pharmacistId) throws PharmacistException, CartException {
      System.out.println("In get of Create Cart Method by Pharmacist
ID");
        Cart cart = cartService.findPharmacistCart(pharmacistId);
        return ResponseEntity.ok(cart);
    }
}
DistributprController :
             ------------------------------------------------------
---------------------------------------------------------- package
com.cdac_project.controller;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.cdac_project.exception.CartMedicineException;
import com.cdac_project.exception.PharmacistException;
import com.cdac_project.model.CartMedicine;
```

```
import com.cdac_project.model.Medicine;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.request.CreateMedicineRequest;
import com.cdac_project.service.CartMedicineService;
import com.cdac_project.service.MedicineService;
import com.cdac_project.service.PharmacistService;

@RestController
@RequestMapping("/cart/medicine")
@CrossOrigin(origins = "http://localhost:3000")
public class CartMedicineController {

    @Autowired
    private final CartMedicineService cartMedicineService;
    @Autowired
    private final MedicineService medicineService;
    @Autowired
    private PharmacistService pharmacistService;

    public CartMedicineController(CartMedicineService
cartMedicineService,MedicineService medicineService) {
        this.cartMedicineService = cartMedicineService;
            this.medicineService = medicineService;
    }


    @PostMapping("/create")
    public ResponseEntity<?> createCartMedicine(@RequestBody
CreateMedicineRequest request) {
        try {
            if (request == null || request.getMedicineId() <= 0 ||
request.getPharmacistId() <= 0 || request.getMedicineQuantity() <= 0)
{
                return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Invalid input
request");
            }

            Pharmacist pharmacist =
pharmacistService.findPharmacistById(request.getPharmacistId());
            if (pharmacist == null) {
                return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Pharmacist not
found with ID: " + request.getPharmacistId());
            }

            Optional<Medicine> optionalMedicine =
medicineService.findById(request.getMedicineId());
            if (!optionalMedicine.isPresent()) {
                return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Medicine not
found with ID: " + request.getMedicineId());
            }

            Medicine medicine = optionalMedicine.get();
            CartMedicine cartMedicine = new CartMedicine();
            cartMedicine.setPharmacistId(request.getPharmacistId());
```

```
            cartMedicine.setQuantity(request.getMedicineQuantity());
            cartMedicine.setMedicine(medicine);
            cartMedicine.setPrice(request.getMedicineQuantity() *
medicine.getUnitPrice());

            CartMedicine createdMedicine =
cartMedicineService.createCartMedicine(cartMedicine);
            return ResponseEntity.ok("Cart Medicine created
successfully");
        } catch (Exception e) {
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
        }
    }



    @PutMapping("/put/{pharmacistId}/{medsId}")
    public ResponseEntity<?> updateCartMedicine(@PathVariable int
pharmacistId,
                                                @PathVariable int
medsId,
                                                @RequestBody
CartMedicine cartMedicine) throws PharmacistException {
      System.out.println("In getting 1 method of CartMedicine");
        try {
            CartMedicine updatedMedicine =
cartMedicineService.updateCartMedicine(pharmacistId, medsId,
cartMedicine);
            return ResponseEntity.ok("Cart Medicine updated
successfully");
        } catch (CartMedicineException e) {
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
        }
    }

    @DeleteMapping("/del/{pharmacistId}/{medsId}")
    public ResponseEntity<?> removeCartMedicine(@PathVariable int
pharmacistId,
                                                @PathVariable int
medsId) throws PharmacistException {
      System.out.println("In getting 2 method of CartMedicine");
        try {
            cartMedicineService.removeCartMedicine(pharmacistId,
medsId);
            return ResponseEntity.ok("Cart Medicine removed
successfully");
        } catch (CartMedicineException e) {
            return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
        }
    }

    private CartMedicine
convertRequestToCartMedicine(CreateMedicineRequest request) throws
CartMedicineException {
```

```
      CartMedicine cartMedicine = new CartMedicine();
        cartMedicine.setPharmacistId(request.getPharmacistId());
        cartMedicine.setQuantity(request.getMedicineQuantity());

        // Fetch the Medicine object from the database based on the
provided medicineId
        Optional<Medicine> optionalMedicine =
medicineService.findById(request.getMedicineId());
        if (optionalMedicine.isPresent()) {
            Medicine medicine = optionalMedicine.get();
            cartMedicine.setMedicine(medicine);
            // Calculate and set the price based on the quantity and
unit price of the medicine
            cartMedicine.setPrice(request.getMedicineQuantity() *
medicine.getUnitPrice());
        } else {
            throw new CartMedicineException("Medicine with ID " +
request.getMedicineId() + " not found");
        }
            return cartMedicine;
    }
}

Distributor :
    ~>
--------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.cdac_project.exception.DistributorException;
import com.cdac_project.model.Distributor;
import com.cdac_project.repository.DistributorRepository;
import com.cdac_project.request.LoginRequest;
import com.cdac_project.response.AuthResponse;
import
com.cdac_project.service.CustomDistributorServiceImplementation;
import com.cdac_project.service.DistributionServiceImplementation;

@RestController
@CrossOrigin(origins = "http://localhost:3000")
@RequestMapping("/distributor")
public class DistributorAuthController {

      @Autowired
    private DistributorRepository distributorRepository;
      @Autowired
      private CustomDistributorServiceImplementation
customDistributorService;
```

```
      @Autowired
      private DistributionServiceImplementation distributorService;

          public DistributorAuthController(DistributorRepository
distributorRepository, CustomDistributorServiceImplementation
customDistributorService) {
              this.distributorRepository = distributorRepository;
              this.customDistributorService =
customDistributorService;
          }

          @PostMapping("/signup")
          public ResponseEntity<AuthResponse>
createUserHandler(@RequestBody Distributor distributor) throws
DistributorException{

             String email = distributor.getEmail();
               String name = distributor.getName();
               String password = distributor.getPassword();

               Distributor isEmailExist =
distributorRepository.findByEmail(email);
               if (isEmailExist != null) {
                   throw new DistributorException("Email is Already
in Use with Another Account! ");
               }
               Distributor createdDistributor = new Distributor();
               createdDistributor.setEmail(email);
               createdDistributor.setName(name);
               createdDistributor.setPassword(password);

               Distributor savedDistributor =
distributorRepository.save(createdDistributor);

               AuthResponse authResponse = new AuthResponse();
               authResponse.setMessage("Sign-Up Success");

               return new ResponseEntity<>(authResponse,
HttpStatus.CREATED);
    }


          @PostMapping("/login")
          public Distributor login(@RequestBody LoginRequest request)
{
              return
distributorService.authenticate(request.getEmail(),
request.getPassword());
          }
}

HomeController :
                ~>
                  Future Scope
 MedicineController :
              ~>
```

```
                ------------------------------------------------------
------------------------------------------------------------- package
com.cdac_project.controller;

import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.cdac_project.exception.*;
import com.cdac_project.model.Medicine;
import com.cdac_project.model.MedicineSearchCriteria;
import com.cdac_project.request.CreateMedicineRequest;
import com.cdac_project.service.MedicineService;

@RestController
@RequestMapping("/medicine")
@CrossOrigin(origins = "http://localhost:3000")
public class MedicineController {

      @Autowired
    private MedicineService medicineService;

      public MedicineController(MedicineService medicineService) {
            this.medicineService = medicineService;
      }


    @PostMapping("/create")
    public Medicine createMedicine(@RequestBody CreateMedicineRequest
req) {
        return medicineService.createMedicine(req);
    }

    @DeleteMapping("/delete/{id}")
    public String deleteMedicine(@PathVariable("id") int id) throws
MedicineException {
      System.out.println("sddddddddddddddd");
        return medicineService.deleteMedicine(id);
    }

    @PutMapping("/update/{id}")
    public Medicine updateMedicine(@PathVariable("id") int id,
@RequestBody Medicine req) throws MedicineException {
        return medicineService.updateMedicine(id, req);
    }

    @GetMapping("/get/{id}")
    public Medicine findMedicineById(@PathVariable("id") int id)
throws MedicineException {
      System.out.println("sddddddddddddddd");
        return medicineService.findMedicineById(id);
    }
```

```
    @GetMapping("/getByCategory/{categoryId}")
    public List<Medicine>
findMedicineByCategory(@PathVariable("categoryId") int categoryId)
throws Exception {
        return medicineService.findMedicineByCategory(categoryId);
    }

    @GetMapping("/search")
    public Page<Medicine> searchMedicines(
            @RequestParam(required = false) String medicineName,
            @RequestParam(required = false) Integer categoryId,
            @RequestParam(required = false) Integer quantity,
            @RequestParam(required = false) Integer price,
            @RequestParam(required = false) LocalDate manufactureDate
    ) throws MedicineException {
        MedicineSearchCriteria criteria = new
MedicineSearchCriteria(medicineName, categoryId, quantity, price,
manufactureDate);
        return medicineService.searchMedicines(criteria);
    }

    @GetMapping("/get-all")
    public ResponseEntity<Page<Medicine>> getAllMedicines(
            @RequestParam(defaultValue = "0") int page,
            @RequestParam(defaultValue = "10") int size
    ) {
        Page<Medicine> medicines =
medicineService.getAllMedicines(page, size);
        return ResponseEntity.ok(medicines);
    }

    @GetMapping("/get-by-id/{id}")
    public ResponseEntity<Medicine>
getMedicineById(@PathVariable("id") int id) {
        Optional<Medicine> medicine = medicineService.findById(id);
        return
medicine.map(ResponseEntity::ok).orElse(ResponseEntity.notFound().bui
ld());
    }



}
OrderController :
       ~>
   ------------------------------------------------------------------
-------------------------------------------- package
com.cdac_project.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.cdac_project.exception.OrderException;
import com.cdac_project.model.Address;
```

```
import com.cdac_project.model.Order;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.service.CartService;
import com.cdac_project.service.OrderService;
import com.cdac_project.service.PharmacistService;

import java.util.List;

@RestController
@RequestMapping("/orders")
@CrossOrigin(origins = "http://localhost:3000")
public class OrderController {


    @Autowired
    private OrderService orderService;

    @Autowired
    private CartService cartService;

    @Autowired
    private PharmacistService userService;


    public OrderController(OrderService orderService) {
        this.orderService = orderService;
    }

    @PostMapping("/create")
    public ResponseEntity<?> createOrder(@RequestBody Pharmacist
user, @RequestBody Address address) throws OrderException {
        Order createdOrder = orderService.createOrder(user, address);
            return ResponseEntity.ok("Order created successfully with
ID: " + createdOrder.getId());
    }

    @GetMapping("/{orderId}")
    public ResponseEntity<?> getOrderById(@PathVariable int orderId)
{
        try {
            Order order = orderService.findOrderByld(orderId);
            return ResponseEntity.ok(order);
        } catch (OrderException e) {
            return
ResponseEntity.status(HttpStatus.NOT_FOUND).body(e.getMessage());
        }
    }

    @GetMapping("/pharmacist/{pharmacistId}")
    public ResponseEntity<?> getPharmacistOrders(@PathVariable int
pharmacistId) {
        List<Order> orders =
orderService.pharmacistOrdersHistory(pharmacistId);
        return ResponseEntity.ok(orders);
    }

    @GetMapping("/all")
```

```
    public ResponseEntity<?> getAllOrders() {
        try {
            List<Order> orders = orderService.getAllOrders();
            return ResponseEntity.ok(orders);
        } catch (OrderException e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Failed
to retrieve orders: " + e.getMessage());
        }
    }

    @PutMapping("/{orderId}")
    public ResponseEntity<?> updateOrder(@PathVariable int orderId,
@RequestBody Order updatedOrder) {
        try {
            Order order = orderService.updateOrder(orderId,
updatedOrder);
            return ResponseEntity.ok(order);
        } catch (OrderException e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Failed
to update order: " + e.getMessage());
        }
    }


    @DeleteMapping("/{orderId}")
    public ResponseEntity<?> deleteOrder(@PathVariable int orderId) {
        try {
            orderService.deleteOrder(orderId);
            return ResponseEntity.ok("Order deleted successfully");
        } catch (OrderException e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Failed
to delete order: " + e.getMessage());
        }
    }
 // Mapping to place an order
    @PostMapping("/place/{orderId}")
    public ResponseEntity<?> placeOrder(@PathVariable int orderId) {
        try {
            Order placedOrder = orderService.placedOrder(orderId);
            return ResponseEntity.ok("Order placed successfully with
ID: " + placedOrder.getId());
        } catch (OrderException e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("Failed to place order: " +
e.getMessage());
        }
    }

 // Mapping to confirm an order
    @PutMapping("/confirm/{orderId}")
    public ResponseEntity<?> confirmOrder(@PathVariable int orderId)
{
        try {
```

```
            Order confirmedOrder =
orderService.confirmedOrder(orderId);
            return ResponseEntity.ok("Order confirmed successfully
with ID: " + confirmedOrder.getId());
        } catch (OrderException e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("Failed to confirm order: " +
e.getMessage());
        }
    }

 // Mapping to ship an order
    @PutMapping("/ship/{orderId}")
    public ResponseEntity<?> shipOrder(@PathVariable int orderId) {
        try {
            Order shippedOrder = orderService.shippedOrder(orderId);
            return ResponseEntity.ok("Order shipped successfully with
ID: " + shippedOrder.getId());
        } catch (OrderException e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("Failed to ship order: " + e.getMessage());
        }
    }

    // Mapping to cancel an order
    @PutMapping("/cancel/{orderId}")
    public ResponseEntity<?> cancelOrder(@PathVariable int orderId) {
        try {
            Order cancelledOrder =
orderService.cancledOrder(orderId);
            return ResponseEntity.ok("Order cancelled successfully
with ID: " + cancelledOrder.getId());
        } catch (OrderException e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body("Failed to cancel order: " +
e.getMessage());
        }
    }
}
PharmacistController :

------------------------------------------------------------------------
------------------------------------------ package
com.cdac_project.controller;



import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.*;
import org.springframework.web.bind.annotation.RestController;
```

```
import com.cdac_project.exception.PharmacistException;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.repository.PharmacistRepository;
import com.cdac_project.request.LoginRequest;
import com.cdac_project.response.AuthResponse;
import
com.cdac_project.service.CustomPharmacistServiceImplementation;
import com.cdac_project.service.PharmacistServiceImplementation;

@RestController
@CrossOrigin(origins = "http://localhost:3000")
@RequestMapping("/pharmacist")
public class ParmacistController {

      @Autowired
    private PharmacistRepository pharmacistRepository;
    @Autowired
    private PharmacistServiceImplementation pharmacistService;
    @Autowired
    private CustomPharmacistServiceImplementation
customPharmacistService;

    public ParmacistController(PharmacistRepository
pharmacistRepository, CustomPharmacistServiceImplementation
customPharmacistService) {
        this.pharmacistRepository = pharmacistRepository;
        this.customPharmacistService = customPharmacistService;
    }

    @PostMapping("/signup")
    public ResponseEntity<AuthResponse>
createUserHandler(@RequestBody Pharmacist pharmacist) throws
PharmacistException {

      System.out.println("In Sign-Up Method of Pharmacist");

        String name = pharmacist.getName();
        String License_Number = pharmacist.getLicenseNumber();
        String email = pharmacist.getEmail();
        String Address = pharmacist.getAddress();
        String password = pharmacist.getPassword();

        Pharmacist isEmailExist =
pharmacistRepository.findByEmail(email);
        if (isEmailExist != null) {
            throw new PharmacistException("Email is Already in Use
with Another Account! ");
        }
        Pharmacist createdPharmacist = new Pharmacist();
        createdPharmacist.setName(name);
        createdPharmacist.setLicenseNumber(License_Number);
        createdPharmacist.setEmail(email);
        createdPharmacist.setAddress(Address);
        createdPharmacist.setPassword(password);

        Pharmacist savedPharmacist =
pharmacistRepository.save(createdPharmacist);
```

```
        AuthResponse authResponse = new AuthResponse();
        authResponse.setMessage("Sign-Up Success");

        return new ResponseEntity<AuthResponse>(authResponse,
HttpStatus.CREATED);
    }

    @PostMapping("/login")
    public Pharmacist login(@RequestBody LoginRequest request) {
        return pharmacistService.authenticate(request.getEmail(),
request.getPassword());
    }

}
```

Exception :~

```
------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.exception;

public class AddressNotFoundException extends Exception

      public AddressNotFoundException(String Message) {
            super(Message);
      }
}
```

```
------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.exception;

public class BillException extends Exception {

      public BillException(String message) {
            super(message);
      }
}
```

```
------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.exception;

public class CartException extends Exception {

      public CartException(String message) {
            super(message);
      }
}
```

```
------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.exception;
```

```
public class CartIsEmptyExcetpion extends Exception {

	public CartIsEmptyExcetpion(String message) {
		super(message);
	}
}
```

```
package com.cdac_project.exception;

public class CartMedicineException extends Exception {

	public CartMedicineException(String message) {
		super(message);
	}
}
```

```
package com.cdac_project.exception;

public class CartNotFoundException extends Exception {

	public CartNotFoundException(String message) {
		super(message);
	}
}
```

```
package com.cdac_project.exception;

public class DistributorException extends Exception {

	public DistributorException(String message) {
		super(message);
	}
}
```

```
package com.cdac_project.exception;

public class MedicineException extends Exception {
	public MedicineException String message) {
		super(message);
	}
}
```

```
package com.cdac_project.exception;

public class OrderException extends Exception {
```

```
	public OrderException(String message) {
		super(message);
	}
}
```

```
package com.cdac_project.exception;

public class PharmacistException extends Exception {

	public PharmacistException(String message) {
		super(message);
	}
}
```

Repositories :

```
package com.cdac_project.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import com.cdac_project.model.Address;

import java.util.List;

@Repository
public interface AddressRepository extends JpaRepository<Address,
Integer> {
    List<Address> findByPharmacistid(int pharmacistId);

//    @Query("SELECT a FROM Address a WHERE a.pharmacistId =
:pharmacistId") // Make sure the property name matches the entity
class
//     List<Address> findByPharmacistid(@Param("pharmacistId") int
pharmacistId);
}
```

```
package com.cdac_project.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
```

```
import com.cdac_project.model.Cart;
import com.cdac_project.model.CartMedicine;
import com.cdac_project.model.Medicine;

@Repository
public interface CartMedicineRepository extends
JpaRepository<CartMedicine, Integer>{

      //CartMedicine isCartMedicineExist(Cart cart, Medicine
medicine, int id);

      boolean existsByCartAndMedicineAndId(Cart cart, Medicine
medicine, int id);

      // In CartMedicineRepository.java
      @Query("SELECT cm FROM cart_medicine cm " +
              "WHERE cm.cart.id = :cartId " +
              "AND cm.medicine.id = :medicineId " +
              "AND cm.pharmacistId = :pharmacistId " +
              "AND cm.quantity = :quantity")
      CartMedicine isCartMedicineExist(@Param("cartId") Cart cart,
                                       @Param("medicineId") Medicine
medicine,
                                       @Param("pharmacistId") int
pharmacistId,
                                       @Param("quantity") int
quantity);


    CartMedicine findByCartAndMedicineAndPharmacistId(Cart cart,
Medicine medicine, int pharmacistId);

}

-------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import com.cdac_project.model.Cart;

public interface CartRepository extends JpaRepository<Cart, Integer>{

      @Query("SELECT c FROM Cart c WHERE c.pharmacist.id=
:pharmacistId")
      public Cart findByPharmacistId(@Param("pharmacistId")int
pharmacistId);

}

-------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;

import com.cdac_project.model.Distributor;

public interface DistributorRepository extends
JpaRepository<Distributor, Integer> {
      public Distributor findByEmail(String distributorEmail);
}

------------------------------------------------------------------------
---------------------------------------------- package
com.cdac_project.repository;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import com.cdac_project.model.MedicineCategory;

public interface MedicineCategoryRepository extends
JpaRepository<MedicineCategory, Integer>{

       public  MedicineCategory findByCategoryid(int categoryId);

//     @Query("Select c from medicine_db c Where
c.Category_id=:Category_id")
//     public MedicineCategory findById
//     (@Param("Category_id") int Category_id);
}

------------------------------------------------------------------------
---------------------------------------------- package
com.cdac_project.repository;

import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import com.cdac_project.model.Medicine;
import com.cdac_project.model.MedicineCategory;

@Repository
public interface MedicineRepository extends JpaRepository<Medicine,
Integer>{


      @Query("SELECT m FROM Medicine m " +
                  "WHERE (:Medicine_Name IS NULL OR m.name =
:Medicine_Name) " +
```

```
                "AND (:Category_id IS NULL OR m.categoryId.id =
:Category_id) " +
                "AND (:Medicine_Quantity IS NULL OR m.quantity <=
:Medicine_Quantity) " +
                "AND (:Manufacture_Date IS NULL OR m.manufactureDate
= :Manufacture_Date) " +
                "AND (:Unit_Price IS NULL OR m.unitPrice =
:Unit_Price)")
          public List<Medicine> filterMedicines(
                  @Param("Medicine_Name") String Medicine_Name,
                  @Param("Category_id") Integer Category_id,
                  @Param("Medicine_Quantity") int Medicine_Quantity,
                  @Param("Manufacture_Date") LocalDate
Manufacture_Date,
                  @Param("Unit_Price") int Unit_Price);

          public List<Medicine> findByCategoryId(int category_id);

}

------------------------------------------------------------------------
---------------------------------------------- package
com.cdac_project.repository;

import java.util.List;

import org.springframework.stereotype.Repository;

import com.cdac_project.model.*;

@Repository
public interface OrderMedicineRepository {
       List<OrderMedicine> findByOrder(Order order);

          List<OrderMedicine> findByMedicine(Medicine medicine);

          List<OrderMedicine> findByOrderAndMedicine(Order order,
Medicine medicine);
}

------------------------------------------------------------------------
---------------------------------------------- package
com.cdac_project.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import com.cdac_project.model.Order;

@Repository
public interface OrderRepository extends JpaRepository<Order,
Integer> {
    List<Order> findByPharmacistId(int pharmacistId);
```

```
    @Query("SELECT o FROM Order o WHERE o.pharmacistId =
:pharmacistID")
    public List<Order> getPharmacistOrders(@Param("pharmacistID") int
id);
}

------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.cdac_project.model.Pharmacist;

public interface PharmacistRepository extends
JpaRepository<Pharmacist, Integer> {
    Pharmacist findByEmail(String Email);
}


------------------------------------------------------------------------
----
```

Request :~

```
------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.request;

public class AddMedicineRequest {

      private int pID;
    private int medicineID;
    private int quantity;
    private int unitPrice;

    public AddMedicineRequest() {
        super();
    }

    public AddMedicineRequest int pID,int medicineID, int quantity,
int unitPrice) {
        super();
        this.pID=pID;
        this.medicineID = medicineID;
        this.quantity = quantity;
        this.unitPrice = unitPrice;
    }

    public int getpID() {
            return pID;
    }

      public void setpID(int pID) {
            this.pID = pID;
    }
```

```
      public int getMedicineID() {
        return medicineID;
    }

    public void setMedicineID(int medicineID) {
        this.medicineID = medicineID;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public int getUnitPrice() {
        return unitPrice;
    }

    public void setUnitPrice(int unitPrice) {
        this.unitPrice = unitPrice;
    }
}
```

```
---------------------------------------------- package
com.cdac_project.request;

import java.time.LocalDate;

import javax.persistence.Column;

public class CreateMedicineRequest {

      // private int Medicine_id;
       private String Medicinename;
       private int Categoryid;
       private int MedicineQuantity;
       private LocalDate ManufactureDate;
       private int UnitPrice;
       private int pharmacistId;
     private int medicineId;



      public int getPharmacistId() {
            return pharmacistId;
      }
      public void setPharmacistId(int pharmacistId) {
            this.pharmacistId = pharmacistId;
      }
      public int getMedicineId() {
            return medicineId;
```

```
	}
	public void setMedicineId(int medicineId) {
		this.medicineId = medicineId;
	}
	public CreateMedicineRequest() {
		super();
	}


	public CreateMedicineRequest(String medicinename, int
categoryid, int medicineQuantity, LocalDate manufactureDate,
			int unitPrice, int pharmacistId, int medicineId) {
		super();
		Medicinename = medicinename;
		Categoryid = categoryid;
		MedicineQuantity = medicineQuantity;
		ManufactureDate = manufactureDate;
		UnitPrice = unitPrice;
		this.pharmacistId = pharmacistId;
		this.medicineId = medicineId;
	}
	public String getMedicinename() {
		return Medicinename;
	}
	public void setMedicinename(String medicine_name) {
		Medicinename = medicine_name;
	}
	public int getCategoryid() {
		return Categoryid;
	}
	public void setCategoryid(int category_id) {
		Categoryid = category_id;
	}
	public int getMedicineQuantity() {
		return MedicineQuantity;
	}
	public void setMedicineQuantity( int medicine_Quantity) {
		MedicineQuantity = medicine_Quantity;
	}
	public LocalDate getManufactureDate() {
		return ManufactureDate;
	}
	public void setManufactureDate(LocalDate manufacture_Date) {
		ManufactureDate = manufacture_Date;
	}
	public int getUnitPrice() {
		return UnitPrice;
	}
	public void setUnitPrice( int unit_Price) {
		UnitPrice = unit_Price;
	}
	/**
	 * @return the category_Name
	 */
//	public String getCategory_Name() {
//		return Category_Name;
//	}
```

```
//    /**
//     * @param category_Name the category_Name to set
//     */
//    public void setCategory_Name(String category_Name) {
//          Category_Name = category_Name;
//    }

}
```

```
package com.cdac_project.request;

public class LoginRequest {

      private String Email;
      private String password;
      public LoginRequest() {
            super();
      }
      public LoginRequest(String Email, String password) {
            super();
            this.Email = Email;
            this.password = password;
      }
      public String getEmail() {
            return Email;
      }
      public void setEmail(String Email) {
            this.Email = Email;
      }
      public String getPassword() {
            return password;
      }
      public void setPassword String password) {
            this.password = password;
      }

}
```

```
package com.cdac_project.request;

import java.time.LocalDate;

public class MedicineRequest {

      private int id;
      private String medicineName;
    private int categoryId;
    private int medicineQuantity;
    private LocalDate manufactureDate;
      public int getId() {
            return id;
      }
```

```
	public void setId(int id) {
		this.id = id;
	}
	public String getMedicineName() {
		return medicineName;
	}
	public void setMedicineName(String medicineName) {
		this.medicineName = medicineName;
	}
	public int getCategoryId() {
		return categoryId;
	}
	public void setCategoryId(int categoryId) {
		this.categoryId = categoryId;
	}
	public int getMedicineQuantity() {
		return medicineQuantity;
	}
	public void setMedicineQuantity(int medicineQuantity) {
		this.medicineQuantity = medicineQuantity;
	}
	public LocalDate getManufactureDate() {
		return manufactureDate;
	}
	public void setManufactureDate(LocalDate manufactureDate) {
		this.manufactureDate = manufactureDate;
	}
	public MedicineRequest() {
		super();
	}
	public MedicineRequest(int id, String medicineName, int
categoryId, int medicineQuantity,
			LocalDate manufactureDate) {
		super();
		this.id = id;
		this.medicineName = medicineName;
		this.categoryId = categoryId;
		this.medicineQuantity = medicineQuantity;
		this.manufactureDate = manufactureDate;
	}



}
```

Response :~

```
-------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.response;

public class AuthResponse {

	private String message;
	public AuthResponse() {
		super();
```

```
	}
	public AuthResponse(String message) {
		super();
		this.message = message;
	}
	public String getMessage() {
		return message;
	}
	public void setMessage(String message) {
		this.message = message;
	}

}
```

```
-------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.response;

public class LoginResponse {

	String message;

	public LoginResponse(String message) {
		super();
		this.message = message;
	}

	public String getMessage() {
		return message;
	}

	public void setMessage(String message) {
		this.message = message;
	}

}
```

```
-------------------------------------------------------------------------
------------------------------------------- package
com.cdac_project.response;

public class MedicineResponse {
	private String message;
	public MedicineResponse() {
		super();
	}
	public MedicineResponse(String message) {
		super();
		this.message = message;
	}
	public String getMessage() {
		return message;
	}
	public void setMessage(String message) {
		this.message = message;
```

}

Service And ServiceImplementation :

```
package com.cdac_project.service;

import java.util.List;

import com.cdac_project.exception.AddressNotFoundException;
import com.cdac_project.model.Address;

public interface AddressService {
   Address addAddress(Address address);

   Address getAddressById(int addressId) throws AddressNotFoundException;

   List<Address> getAddressesByPharmacistid(int pharmacistId);

   Address updateAddress(int addressId, Address address) throws
AddressNotFoundException;

   void deleteAddress(int addressId) throws AddressNotFoundException;
}
```

```
package com.cdac_project.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cdac_project.exception.AddressNotFoundException;
import com.cdac_project.model.Address;
import com.cdac_project.repository.AddressRepository;

import java.util.List;
import java.util.Optional;

@Service
public class AddressServiceImplementation implements AddressService {

        @Autowired
   private AddressRepository addressRepository;
```

```
//    public AddressServiceImplementation(AddressRepository addressRepository) {
//        this.addressRepository = addressRepository;
//    }

    @Override
    public Address addAddress(Address address) {
        return addressRepository.save(address);
    }

    @Override
    public Address getAddressById(int addressId) throws AddressNotFoundException {
//        Optional<Address> optionalAddress = addressRepository.findById(addressId);
//            return optionalAddress.get();

            Optional<Address> optionalAddress = addressRepository.findById(addressId);
        if (optionalAddress.isPresent()) {
            return optionalAddress.get();
        } else {
            throw new AddressNotFoundException("Address not found with ID: " + addressId);
        }
    }

    @Override
    public List<Address> getAddressesByPharmacistid(int pharmacistId) {
        return addressRepository.findByPharmacistid(pharmacistId);
    }

    @Override
    public Address updateAddress(int addressId, Address address) throws
AddressNotFoundException {
//        Optional<Address> optionalAddress = addressRepository.findById(addressId);
//            address.setAddressid(addressId);
//            return addressRepository.save(address);

            Optional<Address> optionalAddress = addressRepository.findById(addressId);
        if (optionalAddress.isPresent()) {
            Address existingAddress = optionalAddress.get();
            existingAddress.setFullAddress(address.getFullAddress()); // Update Full_Address field
            return addressRepository.save(existingAddress);
        } else {
            throw new AddressNotFoundException("Address not found with ID: " + addressId);
        }
    }

    @Override
    public void deleteAddress(int addressId) throws AddressNotFoundException {
        if (addressRepository.existsById(addressId)) {
```

```
            addressRepository.deleteById(addressId);
        } else {
            throw new AddressNotFoundException("Address not found with ID: " + addressId);
        }
    }
}
```

```
package com.cdac_project.service;

import org.springframework.stereotype.Service;

import com.cdac_project.exception.CartException;
import com.cdac_project.exception.CartMedicineException;
import com.cdac_project.exception.MedicineException;
import com.cdac_project.exception.PharmacistException;
import com.cdac_project.model.Cart;
import com.cdac_project.model.CartMedicine;
import com.cdac_project.model.Medicine;

@Service
public interface CartMedicineService {

        public CartMedicine createCartMedicine(CartMedicine cartMedicine)throws
CartMedicineException, PharmacistException;
        public  CartMedicine updateCartMedicine(int PharmacistID ,int medsid ,
CartMedicine cartMedicine) throws CartMedicineException, PharmacistException;
        public void removeCartMedicine(int pharmacistID , int cartMedicineID) throws
CartMedicineException , PharmacistException  ;
        public CartMedicine findCartMedicineByID(int CartMedicineID)throws
CartMedicineException;
        CartMedicine isCartMedicineExist(Cart cart, int PID, Medicine medicine, int quantity)
                        throws CartException, MedicineException, PharmacistException;

}
```

```
package com.cdac_project.service;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cdac_project.exception.CartException;
import com.cdac_project.exception.CartMedicineException;
```

```
import com.cdac_project.exception.MedicineException;
import com.cdac_project.exception.PharmacistException;
import com.cdac_project.model.Cart;
import com.cdac_project.model.CartMedicine;
import com.cdac_project.model.Medicine;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.repository.CartMedicineRepository;
import com.cdac_project.repository.CartRepository;

@Service
public class CartMedicineServiceImplementation implements CartMedicineService{

	@Autowired
	private CartMedicineRepository cartMedicineRepository;
	@Autowired
	private PharmacistService pharmacistService;
	@Autowired
	private CartRepository cartRepository;


	public CartMedicineServiceImplementation(CartMedicineRepository
cartMedicineRepository,
				PharmacistService pharmacistService, CartRepository
cartRepository) {
		super();
		this.cartMedicineRepository = cartMedicineRepository;
		this.pharmacistService = pharmacistService;
		this.cartRepository = cartRepository;
	}

	@Override
	public CartMedicine createCartMedicine(CartMedicine cartMedicine)
				throws CartMedicineException, PharmacistException {
		// TODO Auto-generated method stub
//
//		cartMedicine.setQuantity((int) 1);
//	cartMedicine.setPrice(cartMedicine.getMedicine().getUnitPrice() *
cartMedicine.getQuantity());
//
//	CartMedicine createdCartMedicine = cartMedicineRepository.save(cartMedicine);
//	return createdCartMedicine;
		 try {
		return cartMedicineRepository.save(cartMedicine);
	} catch (Exception e) {
		throw new CartMedicineException("Error creating cart medicine: " +
e.getMessage());
		}
	}
```

```
	@Override
	public CartMedicine isCartMedicineExist(Cart cart, int PID, Medicine medicine ,int
quantity)
				throws CartException, MedicineException, PharmacistException {
		// TODO Auto-generated method stub

		 CartMedicine cm = cartMedicineRepository.isCartMedicineExist(cart,
medicine, PID,quantity);
		return cm;
	}

	@Override
	public void removeCartMedicine(int pharmacistID, int cartMedicineID)
				throws CartMedicineException, PharmacistException {
		// TODO Auto-generated method stub

		CartMedicine cartMedicine = findCartMedicineByID(cartMedicineID);
    Pharmacist pharmacist =
pharmacistService.findPharmacistById(cartMedicine.getPharmacistId());
    Pharmacist reqPharmacist = pharmacistService.findPharmacistById(pharmacistID);
    if (pharmacist.getId().equals(reqPharmacist.getId())) {
      cartMedicineRepository.deleteById(cartMedicineID);
    } else {
      throw new PharmacistException("This Medicine Does not Belong To Your Cart!");
    }
	}

	@Override
	public CartMedicine findCartMedicineByID(int CartMedicineID) throws
CartMedicineException {
		// TODO Auto-generated method stub

		Optional<CartMedicine> opt =
cartMedicineRepository.findById(CartMedicineID);
    if (opt.isPresent()) {
      return opt.get();
    }
    throw new CartMedicineException("Medicine Not Found!");
    }

	@Override
	public CartMedicine updateCartMedicine(int PharmacistID, int medsid, CartMedicine
cartMedicine)
				throws CartMedicineException, PharmacistException {
		 CartMedicine meds = findCartMedicineByID(medsid);
```

```
                Pharmacist p = pharmacistService.findPharmacistById(PharmacistID);

                if (p.getId().equals(PharmacistID)) {
                   meds.setQuantity(meds.getQuantity());
                   meds.setPrice(meds.getPrice() * meds.getMedicine().getUnitPrice());
                }
                return cartMedicineRepository.save(meds);
             }

}
```

```
package com.cdac_project.service;

import org.springframework.stereotype.Service;

import com.cdac_project.exception.*;
import com.cdac_project.model.*;
import com.cdac_project.request.AddMedicineRequest;

@Service
public interface CartService {

        public Cart createCart(int pharmacist) throws PharmacistException;
        public Cart createCart(Pharmacist pharmacist);
        public String addCartMedicine(int Pharmacistid, int req)throws MedicineException,
CartException, CartMedicineException, PharmacistException ;
   public String addCartMedicine(int Pharmacistid, AddMedicineRequest req)throws
MedicineException, CartException, CartMedicineException, PharmacistException ;
   public Cart findPharmacistCart(int p_ID)  throws CartException;
}
```

```
package com.cdac_project.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cdac_project.exception.CartException;
import com.cdac_project.exception.CartIsEmptyExcetpion;
import com.cdac_project.exception.CartMedicineException;
import com.cdac_project.exception.MedicineException;
import com.cdac_project.exception.PharmacistException;
import com.cdac_project.model.Cart;
import com.cdac_project.model.CartMedicine;
import com.cdac_project.model.Medicine;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.repository.CartRepository;
```

```
import com.cdac_project.request.AddMedicineRequest;

@Service
public class CartServiceImplementation implements CartService {

	@Autowired
	private CartRepository cartRepository;
	@Autowired
	private CartMedicineService cartMedicineService;
	@Autowired
	private MedicineService medicineService;
	@Autowired
	private PharmacistService pharmacistService;

	public CartServiceImplementation(CartRepository cartRepository,
CartMedicineService cartMedicineService,
				MedicineService medicineService) {
		super();
		this.cartRepository = cartRepository;
		this.cartMedicineService = cartMedicineService;
		this.medicineService = medicineService;
	}

	@Override
	public Cart createCart(Pharmacist pharmacist) {
		// TODO Auto-generated method stub

		Cart cart = new Cart();
		cart.setPharmacist(pharmacist);
		return cartRepository.save(cart);
	}

	@Override
	public String addCartMedicine(int Pharmacistid, AddMedicineRequest req) throws
MedicineException, CartException, CartMedicineException, PharmacistException {
		// TODO Auto-generated method stub

		Cart cart = cartRepository.findByPharmacistId(Pharmacistid);
		Medicine medicine =
medicineService.findMedicineById(req.getMedicineID());
		int quantity = medicine.getQuantity();
		CartMedicine isPresent = cartMedicineService.isCartMedicineExist(cart,
Pharmacistid, medicine,quantity);

		if(isPresent==null) {
			CartMedicine cartMedicine = new CartMedicine();
			cartMedicine.setMedicine(medicine);
			cartMedicine.setCart(cart);
```

```
                        cartMedicine.setQuantity(req.getQuantity());
                        cartMedicine.setPharmacistId(Pharmacistid);
                        int price = req.getQuantity()*medicine.getUnitPrice();
                        cartMedicine.setPrice(price);
                        cartMedicine.setQuantity(req.getQuantity());

                        CartMedicine createdCartMedicine =
cartMedicineService.createCartMedicine(cartMedicine);
                        cart.getCartMedicine().add(createdCartMedicine);
                }
                return "Item added to Cart !";
        }

        @Override
        public Cart findPharmacistCart(int p_ID) throws  CartException {
                // TODO Auto-generated method stub
                try {
            Cart cart = cartRepository.findByPharmacistId(p_ID);
            if (cart == null) {
               System.out.println("Cart is currently empty");
               return null;
            }
            int totalPrice = 0;
            int totalMedicine = 0;
            for(CartMedicine cartMedicine : cart.getCartMedicine()) {
               totalPrice += cartMedicine.getPrice();
               totalMedicine += cartMedicine.getQuantity();
            }
            cart.setTotalItem(totalMedicine);
            cart.setTotalPrice(totalPrice);
            cartRepository.save(cart);
            return cart;
          } catch (Exception e) {
            // Handle the exception, log it, and return an appropriate response
            System.out.println("Multiple carts found for pharmacist ID: " + p_ID);
            return null;
          }
        }

        @Override
        public Cart createCart(int pharmacistId) throws PharmacistException {
                    Pharmacist pharmacist =
pharmacistService.findPharmacistById(pharmacistId);
                    if (pharmacist == null) {
                       throw new PharmacistException("Pharmacist not found with ID: " +
pharmacistId);
                    }
                    Cart cart = new Cart();
```

```
                cart.setPharmacist(pharmacist);
                return cartRepository.save(cart);
            }


        @Override
        public String addCartMedicine(int Pharmacistid, int medicineId)
                        throws MedicineException, CartException, CartMedicineException,
PharmacistException {
                Cart cart = cartRepository.findByPharmacistId(Pharmacistid);
            Medicine medicine = medicineService.findMedicineById(medicineId);
            CartMedicine cartMedicine = new CartMedicine();
            cartMedicine.setMedicine(medicine);
            cartMedicine.setCart(cart);
            CartMedicine createdCartMedicine =
cartMedicineService.createCartMedicine(cartMedicine);
            return "Item added to Cart!";
        }


}
```

```
package com.cdac_project.service;

import java.util.*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cdac_project.exception.DistributorException;
import com.cdac_project.model.Distributor;
import com.cdac_project.repository.DistributorRepository;
import com.cdac_project.request.LoginRequest;

@Service
public class CustomDistributorServiceImplementation implements DistributorService{
        @Autowired
        private DistributorRepository distributorRepository;

        public CustomDistributorServiceImplementation(DistributorRepository
distributorRepository)throws DistributorException{
                this.distributorRepository=distributorRepository;
        }
```

```
	@Override
	public Distributor findDistributorById(int Distributor_ID) throws DistributorException
{
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public Distributor addDistributor(Distributor d) throws DistributorException {
		// TODO Auto-generated method stub
		return null;
	}

	@Override
	public boolean login(LoginRequest login) {
		// TODO Auto-generated method stub
		return false;
	}

	@Override
	public Distributor authenticate(String Email, String password) {
		// TODO Auto-generated method stub
		return null;
	}

//	@Override
//	public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
//		// TODO Auto-generated method stub
//
//		Distributor distributor=distributorRepository.findByEmail(username);
//		if(distributor==null) {
//			throw new DistributorException("User with email -"+username+"
not Found! ");
//		}
//		List<GrantedAuthority> authorities =new ArrayList<>();
//		return new
org.springframework.security.core.userdetails.User(distributor.getEmail(),distributor.getPass
word(),authorities);
//	}


}


---------------------------------------------------------------------------------------------------------------
package com.cdac_project.service;
```

```
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.cdac_project.repository.PharmacistRepository;

@Service
public class CustomPharmacistServiceImplementation{

	@Autowired
	private PharmacistRepository pharmacistRepository;

	public CustomPharmacistServiceImplementation(PharmacistRepository
pharmacistRepository){
		this.pharmacistRepository=pharmacistRepository;
	}

//	@Override
//	public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
//		// TODO Auto-generated method stub
//
//		Pharmacist pharmacist=pharmacistRepository.findByEmail(username);
//		if(pharmacist==null) {
//			throw new UsernameNotFoundException("User with email -
"+username+" not Found! ");
//		}
//		List<GrantedAuthority> authorities =new ArrayList<>();
//		return new
org.springframework.security.core.userdetails.User(pharmacist.getEmail(),pharmacist.getPas
sword(),authorities);
//	}


}
```

```
package com.cdac_project.service;

import org.springframework.stereotype.Service;

import com.cdac_project.exception.DistributorException;
import com.cdac_project.exception.PharmacistException;
import com.cdac_project.model.Distributor;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.request.LoginRequest;

@Service
```

```
public interface DistributorService{
	public  Distributor findDistributorById(int Distributor_ID)throws
DistributorException;
	public  Distributor addDistributor(Distributor d)throws DistributorException;
	public boolean login(LoginRequest login);
	public Distributor authenticate(String Email, String password);
}
```

```
package com.cdac_project.service;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cdac_project.exception.DistributorException;
import com.cdac_project.model.Distributor;
import com.cdac_project.repository.DistributorRepository;
import com.cdac_project.request.LoginRequest;

@Service
public class DistributionServiceImplementation implements DistributorService {

	@Autowired
	private DistributorRepository distributorRepository;


	public DistributionServiceImplementation() {
		super();
	}

	public DistributionServiceImplementation(DistributorRepository
distributorRepository) {
		super();
		this.distributorRepository = distributorRepository;
	}

	@Override
	public Distributor findDistributorById(int Distributor_ID) throws DistributorException
{
		// TODO Auto-generated method stub

		Optional<Distributor> distributor =
distributorRepository.findById(Distributor_ID);
		if(distributor.isPresent()) {
			return distributor.get();
		}
```

```
                throw new DistributorException("Distributor Not Found With ID :
"+Distributor_ID);

        }
        @Override
        public Distributor addDistributor(Distributor d) throws DistributorException {
                // TODO Auto-generated method stub
                return distributorRepository.save(d);
        }

        @Override
        public boolean login(LoginRequest login) {
                Distributor p = distributorRepository.findByEmail(login.getEmail());
                return p != null && validatePassword(login.getPassword(), p.getPassword());
          }

        @Override
        public Distributor authenticate(String Email, String password) {
                // TODO Auto-generated method stub
                Distributor user = distributorRepository.findByEmail(Email);
    if (user != null && validatePassword(password, user.getPassword())) {
       return user;
    }
    return null;
        }
         private boolean validatePassword(String inputPassword, String storedPassword) {
              return inputPassword.equals(storedPassword);
        }

}


--------------------------------------------------------------------------
------------------------------------------ package
com.cdac_project.service;

public interface MedicineCategoryService {

}


--------------------------------------------------------------------------
------------------------------------------ package
com.cdac_project.service;

public class MedicineCategoryServiceImplementation implements
MedicineCategoryService{
```

```
package com.cdac_project.service;

import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

import org.springframework.data.domain.Page;
import org.springframework.stereotype.Service;

import com.cdac_project.exception.MedicineException;
import com.cdac_project.model.Medicine;
import com.cdac_project.model.MedicineCategory;
import com.cdac_project.model.MedicineSearchCriteria;
import com.cdac_project.request.CreateMedicineRequest;

@Service
public interface MedicineService {

	public Medicine createMedicine(CreateMedicineRequest req);
	public String deleteMedicine(int Medicine_ID)throws MedicineException;
	public Medicine updateMedicine(int Medicine_ID,Medicine req) throws
MedicineException;
	public Medicine findMedicineById(int id) throws MedicineException;
	public List<Medicine> findMedicineByCategory(int Category_id) throws Exception;
	public Page<Medicine> getAllMedicine(int Medicineid,String MedicineName ,int
CategoryId ,int Quantity, LocalDate ManufactureDate , int UnitPrice) throws
MedicineException;
	public Optional<Medicine> findById(int id);
   public Page<Medicine> searchMedicines(MedicineSearchCriteria criteria);
	public Page<Medicine> getAllMedicines(int page, int size);
}
```

```
package com.cdac_project.service;

import java.time.LocalDate;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
```

```
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

import com.cdac_project.exception.MedicineException;
import com.cdac_project.model.Medicine;
import com.cdac_project.model.MedicineCategory;
import com.cdac_project.model.MedicineSearchCriteria;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.repository.MedicineCategoryRepository;
import com.cdac_project.repository.MedicineRepository;
import com.cdac_project.repository.PharmacistRepository;
import com.cdac_project.request.CreateMedicineRequest;

@Service
public class MedicineServiceImplementation implements MedicineService{

	@Autowired
	private MedicineRepository medicineRepository;
	@Autowired
	private MedicineCategoryRepository categoryRepository;

	public MedicineServiceImplementation() {}

	@Override
	public Medicine createMedicine(CreateMedicineRequest req) {




			System.out.println("Create Medicine");
			  MedicineCategory category =
categoryRepository.findByCategoryid(req.getCategoryid());
		if (category == null) {
				   category = getDefaultCategory();
				  }
				  Medicine medicine = new Medicine();
				  medicine.setName(req.getMedicinename());
				  medicine.setCategoryId(category);
				  medicine.setQuantity(req.getMedicineQuantity());
				  medicine.setManufactureDate(req.getManufactureDate());
				  medicine.setUnitPrice(req.getUnitPrice());
		Medicine savedMedicine = medicineRepository.save(medicine);
				  return savedMedicine;
	  }

	  private MedicineCategory getDefaultCategory() {
		 MedicineCategory defaultCategory = new MedicineCategory();
```

```
            defaultCategory.setCategoryid(1); // Set a default category ID
            return categoryRepository.save(defaultCategory);
        }

        @Override
        public String deleteMedicine(int Medicine_ID) throws MedicineException {
                System.out.println("Delete Medicine");
                 Medicine medicine = findMedicineById(Medicine_ID);
            medicineRepository.delete(medicine);
            return "Medicine Deleted Successfully!";
        }

        @Override
        public Medicine updateMedicine(int Medicine_ID, Medicine req) throws
MedicineException {
                System.out.println("Update Medicine");
                Medicine medicine= findMedicineById(Medicine_ID);
                if(req.getQuantity()!=0) {
                        medicine.setQuantity(req.getQuantity());
                }
                return medicineRepository.save(medicine);
        }

        @Override
        public Medicine findMedicineById(int Medicine_ID) throws MedicineException {
                // TODO Auto-generated method stub
                System.out.println("find by id");
                Optional<Medicine> medicine=medicineRepository.findById(Medicine_ID);
                return medicine.get();
        }


        @Override
        public List<Medicine> findMedicineByCategory(int Category_id) throws Exception{
                System.out.println("find by Category");
                 List<Medicine> medicines = medicineRepository.findAll();
            List<Medicine> filteredMedicines = medicines.stream()
                    .filter(medicine -> medicine.getCategoryId().getCategoryid() ==
Category_id)
                    .collect(Collectors.toList());
              if (!filteredMedicines.isEmpty()) {
                 return filteredMedicines;
              }
              throw new MedicineException("Medicine not Found with Category-ID : " +
Category_id);
        }
```

```
	@Override
	public Page<Medicine> getAllMedicine(int Medicineid, String MedicineName, int
CategoryId, int Quantity,
				LocalDate ManufactureDate, int UnitPrice) throws
MedicineException {
		// TODO Auto-generated method stub
		System.out.println("Get All Medicine");
		 Pageable pageable = PageRequest.of(0, 10); // You can change the page size
and number as needed
		 Page<Medicine> medicines = medicineRepository.findAll(pageable);

		   return medicines;
	}


	@Override
	public Page<Medicine> searchMedicines(MedicineSearchCriteria criteria) {
		// TODO Auto-generated method stub

		System.out.println("Search Medicine");
		return medicineRepository.findAll(PageRequest.of(0, 10));
	}


	@Override
	public Optional<Medicine> findById(int id) {
      return medicineRepository.findById(id);
	}

	@Override
	public Page<Medicine> getAllMedicines(int page, int size) {
		 Pageable pageable = PageRequest.of(page, size);
	      return medicineRepository.findAll(pageable);
	}

}
```

```
 package com.cdac_project.service;

import java.util.List;

import org.springframework.stereotype.Service;

import com.cdac_project.exception.OrderException;
import com.cdac_project.model.Address;
import com.cdac_project.model.Order;
```

```
import com.cdac_project.model.Pharmacist;

@Service
public interface OrderService {

	public Order createOrder(Pharmacist user, Address shippingAdress);//User
	public Order findOrderByld (int orderld) throws OrderException;//User
	public List<Order> pharmacistOrdersHistory (int pharmacistld);//Dist and User
	public Order placedOrder(int orderld) throws OrderException;//User
	public Order confirmedOrder(int orderld) throws OrderException;//Dist
	public Order shippedOrder(int orderld) throws OrderException;//Dist
	public Order deliveredOrder(int orderld) throws OrderException;//Dist
	public Order cancledOrder(int orderld) throws OrderException;//Dist and User
	public List<Order> getAllOrders() throws OrderException;//User and Dist
	public void deleteOrder(int Order_ID) throws OrderException;//User
	public Order updateOrder(int orderId, Order updatedOrder) throws OrderException;


}

-----------------------------------------------------------------------------------------------------------------
package com.cdac_project.service;

import java.time.LocalDateTime;
import java.util.List;
import java.util.Optional;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cdac_project.exception.OrderException;
import com.cdac_project.model.Address;
import com.cdac_project.model.Order;
import com.cdac_project.model.OrderStatus;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.repository.CartRepository;
import com.cdac_project.repository.OrderRepository;

@Service
public class OrderServiceImplementation implements OrderService{

	@Autowired
	private CartRepository cartRepository;
	@Autowired
	private CartMedicineService cartMedicineService;
	@Autowired
```

```
        private MedicineService medicineService;
        @Autowired
        private OrderRepository orderRepository;

        public OrderServiceImplementation() {
    // Default constructor
  }
        public OrderServiceImplementation(CartRepository cartRepository,
CartMedicineService cartMedicineService,
                        MedicineService medicineService) {
                super();
                this.cartRepository = cartRepository;
                this.cartMedicineService = cartMedicineService;
                this.medicineService = medicineService;
        }
        // Constructor injection
  public OrderServiceImplementation(CartMedicineService cartMedicineService) {
    this.cartMedicineService = cartMedicineService;
  }
        @Override
        public Order createOrder(Pharmacist user, Address shippingAdress) {
    Order order = new Order();
    order.setPharmacist(user);
    order.setShippingAddress(shippingAdress);
    order.setOrderDate(LocalDateTime.now());
    // Set other necessary fields
    return orderRepository.save(order);
        }

        @Override
        public Order findOrderById(int orderId) throws OrderException {
    return orderRepository.findById(orderId)
         .orElseThrow(() -> new OrderException("Order not found with ID: " + orderId));
        }

        @Override
        public List<Order> pharmacistOrdersHistory(int pharmacistId) {

    return orderRepository.findByPharmacistId(pharmacistId);
        }

        @Override
  @Transactional
        public Order placedOrder(int orderId) throws OrderException {
                Order order = findOrderById(orderId);
            if (order == null) {
               throw new OrderException("Order not found with ID: " + orderId);
            }
```

```
            order.setStatus(OrderStatus.PLACED);
            // Perform any other necessary operations
            return order;
        }

        @Override
    @Transactional
        public Order confirmedOrder(int orderId) throws OrderException {
                Order order = findOrderByld(orderId);
            if (order == null) {
               throw new OrderException("Order not found with ID: " + orderId);
            }
            order.setStatus(OrderStatus.CONFIRMED);
            // Perform any other necessary operations
            return order;
        }

        @Override
    @Transactional
        public Order shippedOrder(int orderId) throws OrderException {
                // TODO Auto-generated method stub
                Order order = findOrderByld(orderId);
            if (order == null) {
               throw new OrderException("Order not found with ID: " + orderId);
            }
            order.setStatus(OrderStatus.SHIPPED);
            // Set delivery date, if applicable
            // order.setDeliveryDate(LocalDateTime.now());
            // Perform any other necessary operations
            return order;
        }

        @Override
    @Transactional
        public Order deliveredOrder(int orderId) throws OrderException {
                Order order = findOrderByld(orderId);
            if (order == null) {
               throw new OrderException("Order not found with ID: " + orderId);
            }
            order.setStatus(OrderStatus.DELIVERED);
            // Set delivery date, if applicable
            // order.setDeliveryDate(LocalDateTime.now());
            // Perform any other necessary operations
            return order;
        }

        @Override
    @Transactional
```

```
    public Order cancledOrder(int orderId) throws OrderException {
            Order order = findOrderById(orderId);
        if (order == null) {
           throw new OrderException("Order not found with ID: " + orderId);
        }
        order.setStatus(OrderStatus.CANCELED);
        // Perform any other necessary operations
        return order;
    }

    @Override
    public List<Order> getAllOrders() throws OrderException {
            // TODO Auto-generated method stub
            return orderRepository.findAll();
    }

    @Override
    public void deleteOrder(int Order_ID) throws OrderException {
            // TODO Auto-generated method stub
            Optional<Order> optionalOrder = orderRepository.findById(Order_ID);
      if (optionalOrder.isPresent()) {
        orderRepository.deleteById(Order_ID);
      } else {
        throw new OrderException("Order not found with ID: " + Order_ID);
      }
    }

    @Override
    public Order updateOrder(int orderId, Order updatedOrder) throws OrderException {
      Optional<Order> optionalOrder = orderRepository.findById(orderId);
      if (optionalOrder.isPresent()) {
        Order existingOrder = optionalOrder.get();
        // Update fields of existingOrder with corresponding fields from updatedOrder
        // For example:
        existingOrder.setPrice(updatedOrder.getPrice());
        existingOrder.setOrderDate(updatedOrder.getOrderDate());
        // Similarly, update other fields as needed

        return orderRepository.save(existingOrder);
      } else {
        throw new OrderException("Order not found with ID: " + orderId);
      }
    }


}
```

```
package com.cdac_project.service;

import org.springframework.stereotype.Service;

import com.cdac_project.exception.PharmacistException;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.request.LoginRequest;

@Service
public interface PharmacistService{
	public  Pharmacist addPharmacist(Pharmacist p)throws PharmacistException;
	public  Pharmacist findPharmacistById(int p_ID)throws PharmacistException;
	public boolean login(LoginRequest login);
	public Pharmacist authenticate(String Emaail, String password);
}
```

```
package com.cdac_project.service;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.cdac_project.exception.PharmacistException;
import com.cdac_project.model.Pharmacist;
import com.cdac_project.repository.PharmacistRepository;
import com.cdac_project.request.LoginRequest;

@Service
public class PharmacistServiceImplementation implements PharmacistService{

	@Autowired
	private PharmacistRepository pharmacistRepository;

	public PharmacistServiceImplementation() {
		super();
	}

	public PharmacistServiceImplementation(PharmacistRepository
pharmacistRepository) {
		super();
		this.pharmacistRepository = pharmacistRepository;
	}
```

```
	@Override
	public Pharmacist findPharmacistById(int Id) throws PharmacistException {
		// TODO Auto-generated method stub

		Optional<Pharmacist> pharmacist = pharmacistRepository.findById(Id);
		if(pharmacist.isPresent()) {
			return pharmacist.get();
		}
		throw new PharmacistException("Pharmacist Not Found With ID : "+Id);
	}



	@Override
	public Pharmacist addPharmacist(Pharmacist p) throws PharmacistException {
		// TODO Auto-generated method stub
		return pharmacistRepository.save(p);
	}

	@Override
	public boolean login(LoginRequest login) {
	   Pharmacist p = pharmacistRepository.findByEmail(login.getEmail());
	   if (p != null && login.getPassword().equals(p.getPassword())) {
	      return true;
	   }
	   return false;
	}

	@Override
	public Pharmacist authenticate(String Email, String password) {
		// TODO Auto-generated method stub
		Pharmacist user = pharmacistRepository.findByEmail(Email);
    if (user != null && validatePassword(password, user.getPassword())) {
      return user;
    }
    return null;
	}
	 private boolean validatePassword(String inputPassword, String storedPassword) {
		return inputPassword.equals(storedPassword);
	   }

}
```