**WEEK 3**

**Spring Core and Maven, Spring Data JPA with Spring Boot, Hibernate**

**Name: Tadepalli Tarun**

**Superset ID:- 4991850**

***Spring Core and Maven***

1. **Configuring a Basic Spring Application:**

Create a Maven Project LMS

**Add Spring Dependency in Pom.xml:**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

**Create applicationContext.xml:**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id=*"bookRepository"* class=*"com.cognizant.lms.repo.BookRepository"* />

<bean id=*"bookService"* class=*"com.cognizant.lms.service.BookService"*>

<property name=*"bookRepository"* ref=*"bookRepository"* />

</bean>

</beans>

**Create BookRepository.java:**

**package** com.cognizant.lms.repo;

**public** **class** BookRepository {

**public** **void** saveBook(String bookName) {

System.***out***.println("Book '" + bookName + "' saved to the repository.");

}

}

**Create BookService.java:**

**package** com.cognizant.lms.service;

**import** com.cognizant.lms.repo.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for dependency injection

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String bookName) {

System.***out***.println("Adding book: " + bookName);

bookRepository.saveBook(bookName);

}

}

**Create App.java:**

**package** com.cognizant.lms;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.cognizant.lms.service.BookService;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("Spring in Action");

}

}

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Implementing Dependency Injection:**

Create a Maven Project LMS

**Add Spring Dependency in Pom.xml:**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

**Create applicationContext.xml:**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<!-- BookRepository Bean -->

<bean id=*"bookRepository"* class=*"com.cognizant.lms.repo.BookRepository"* />

<!-- BookService Bean with Dependency Injection -->

<bean id=*"bookService"* class=*"com.cognizant.lms.service.BookService"*>

<property name=*"bookRepository"* ref=*"bookRepository"* />

</bean>

</beans>

**Create BookRepository.java:**

**package** com.cognizant.lms.repo;

**public** **class** BookRepository {

**public** **void** saveBook(String bookName) {

System.***out***.println("Book '" + bookName + "' saved to the repository.");

}

}

**Create BookService.java:**

**package** com.cognizant.lms.service;

**import** com.cognizant.lms.repo.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

// Setter for Dependency Injection

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** **void** addBook(String bookName) {

System.***out***.println("BookService: Adding book: " + bookName);

bookRepository.saveBook(bookName);

}

}

**Create App.java:**

**package** com.cognizant.lms;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.cognizant.lms.service.BookService;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("Spring in Action");

}

}

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Creating and Configuring a Maven Project**

**Create a maven project Library Management.**

**Add Spring Dependencies in Pom.xml :**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Servlet API (provided by container like Tomcat) -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

**Configure Maven Compiler Plugin:**

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.10.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

**Output:**

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

***Spring Data JPA with Spring Boot, Hibernate***

1. **Spring Data JPA - Quick Example:**

**Execution:**

**Steps :-**

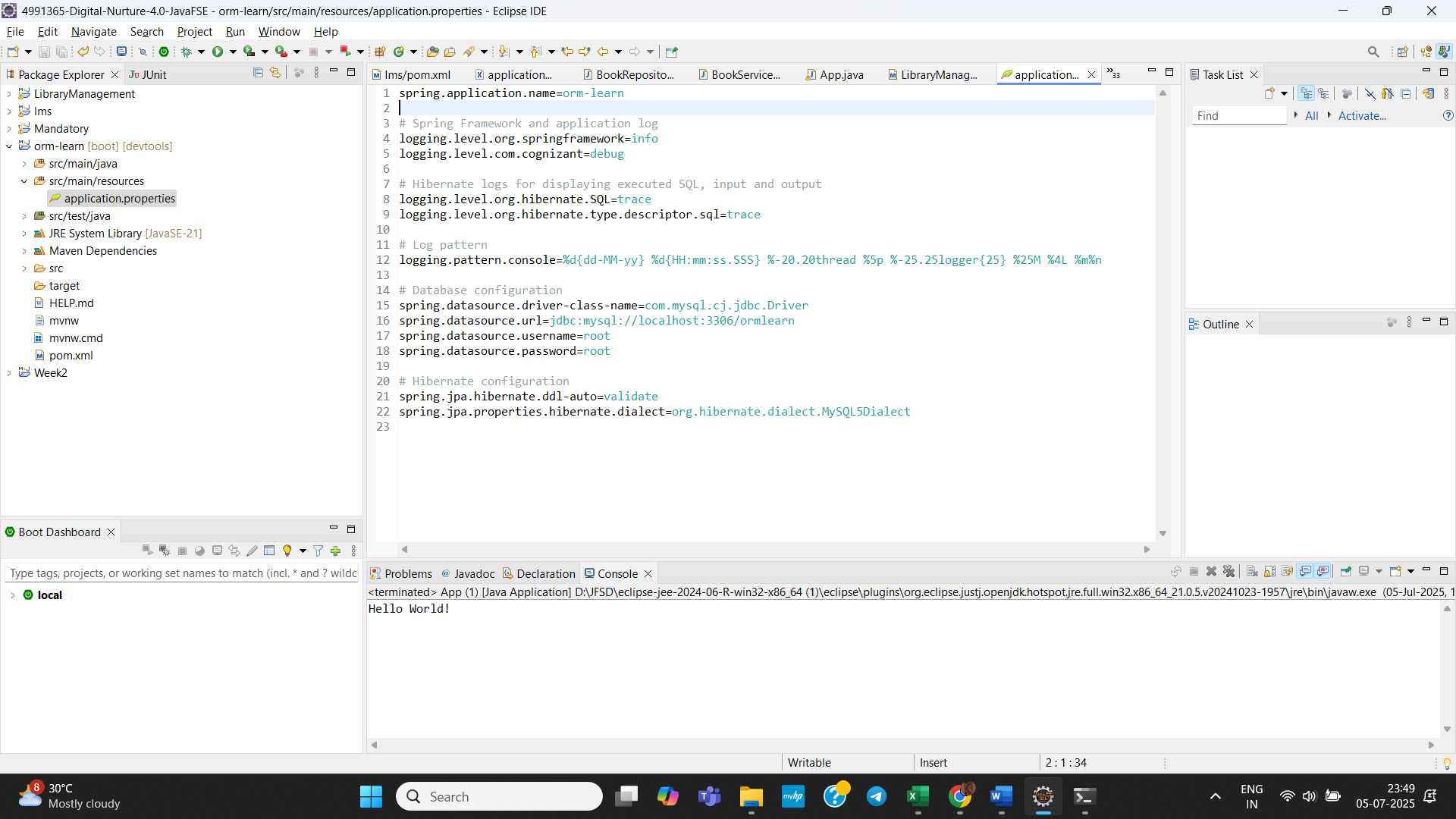
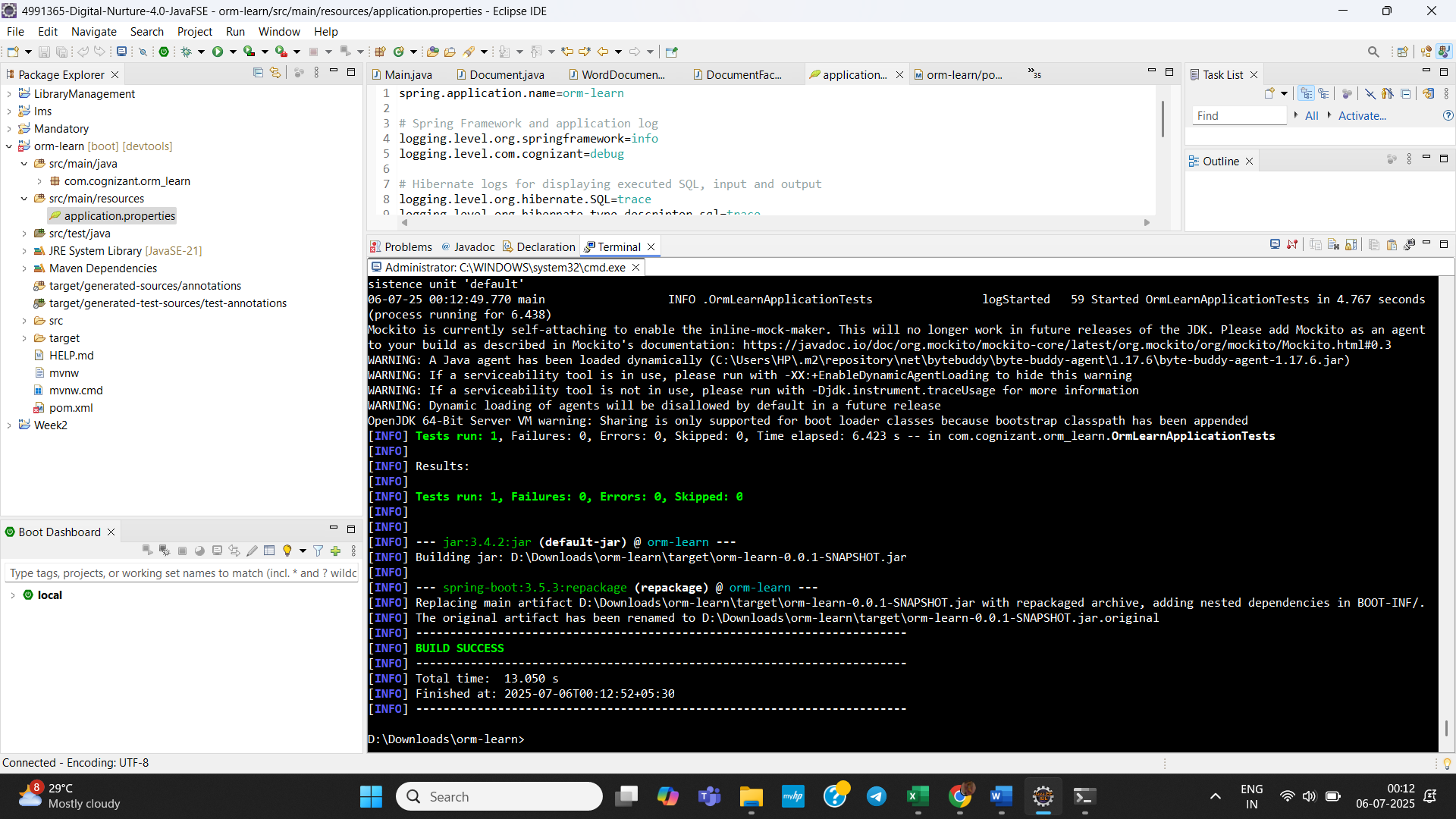
A screenshot of a computer

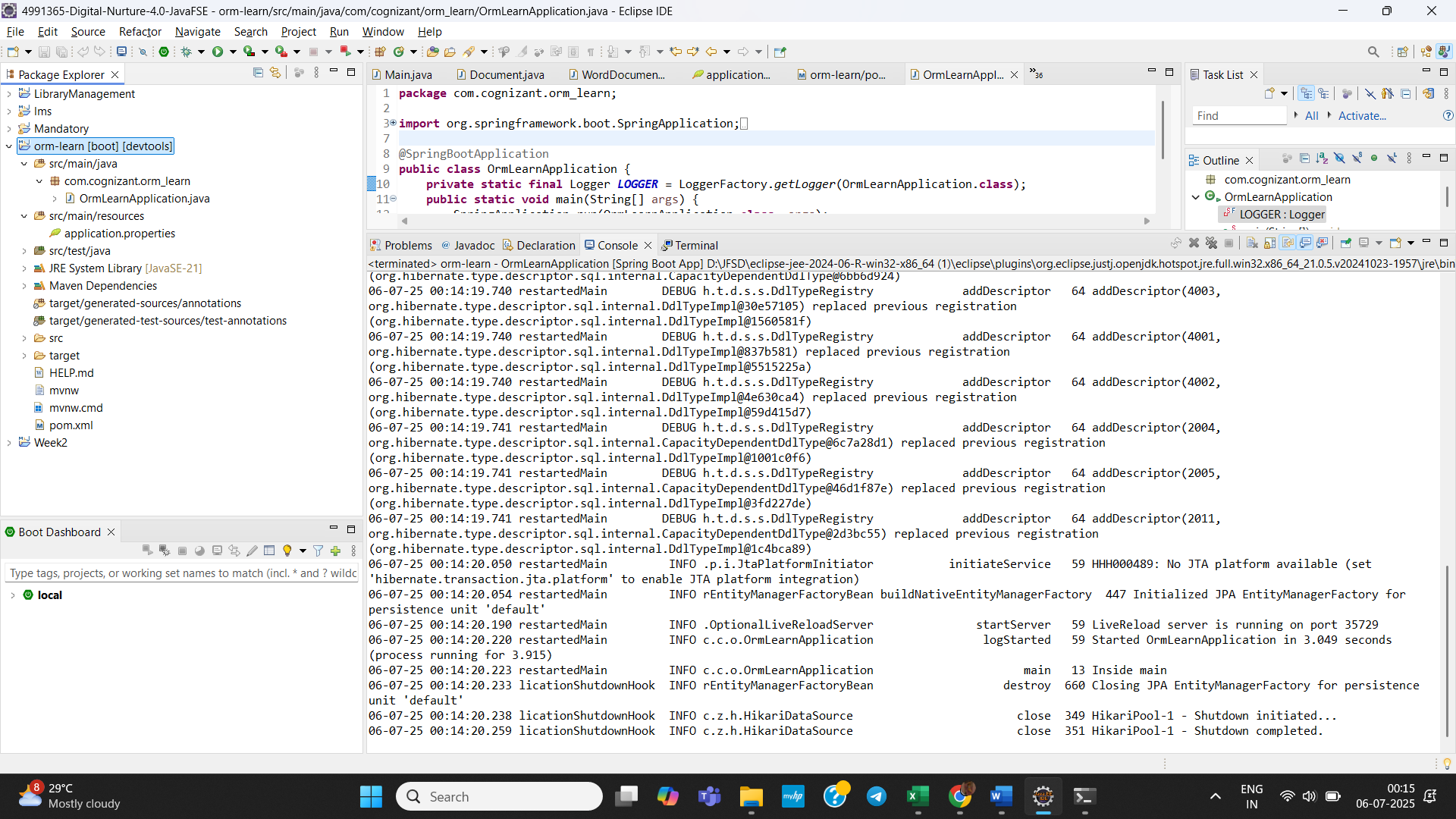
AI-generated content may be incorrect. A screenshot of a computer

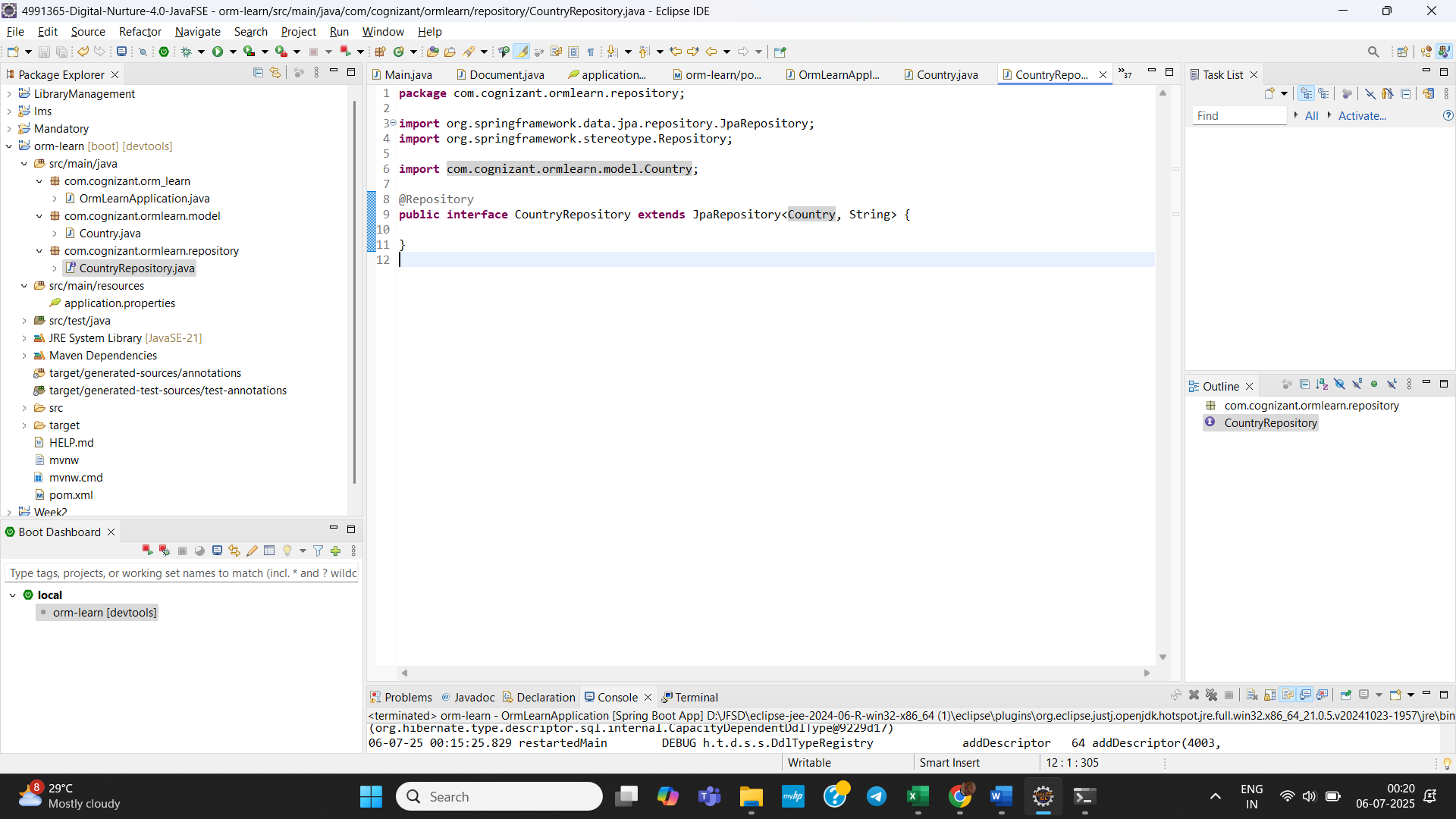
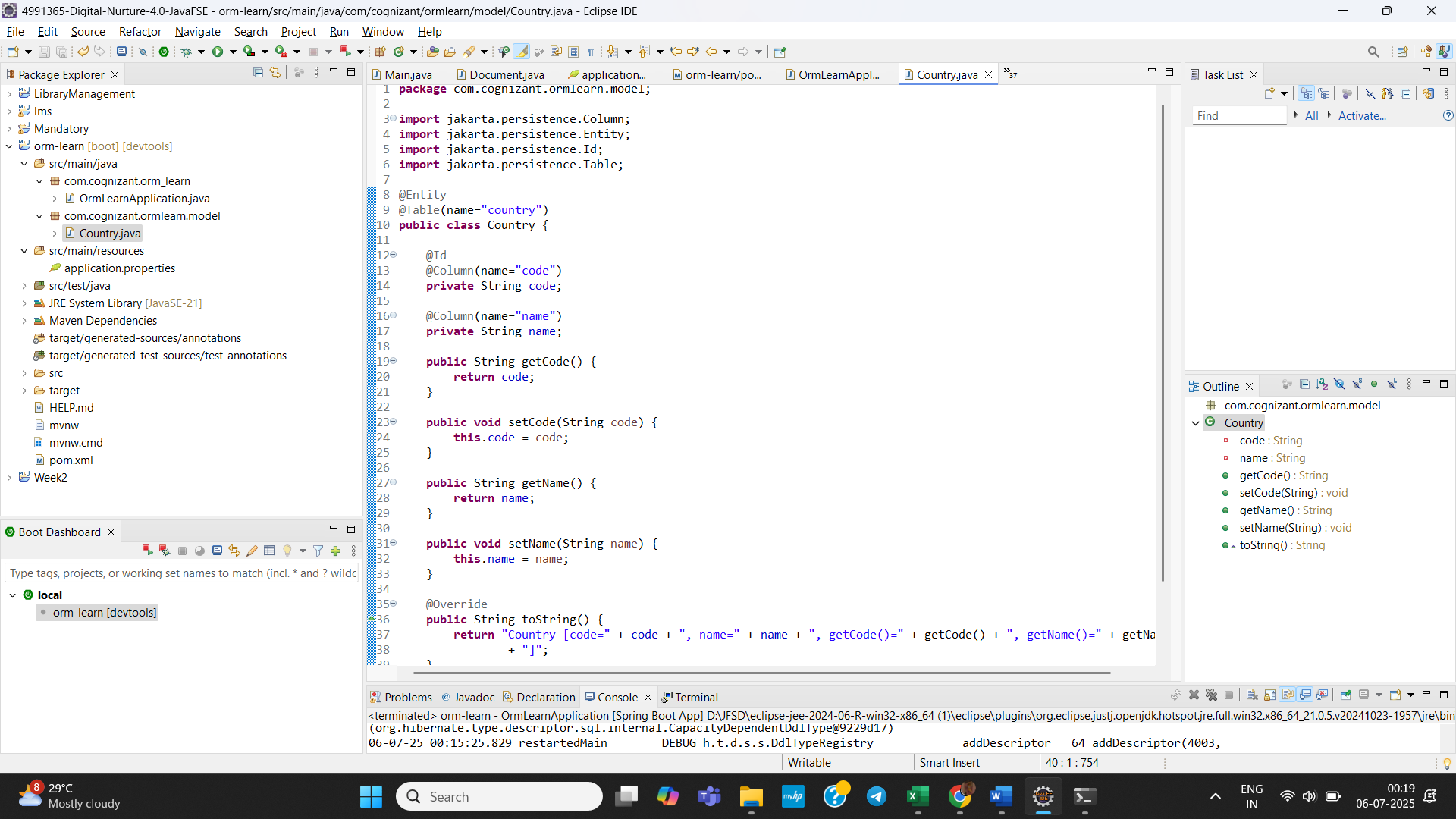
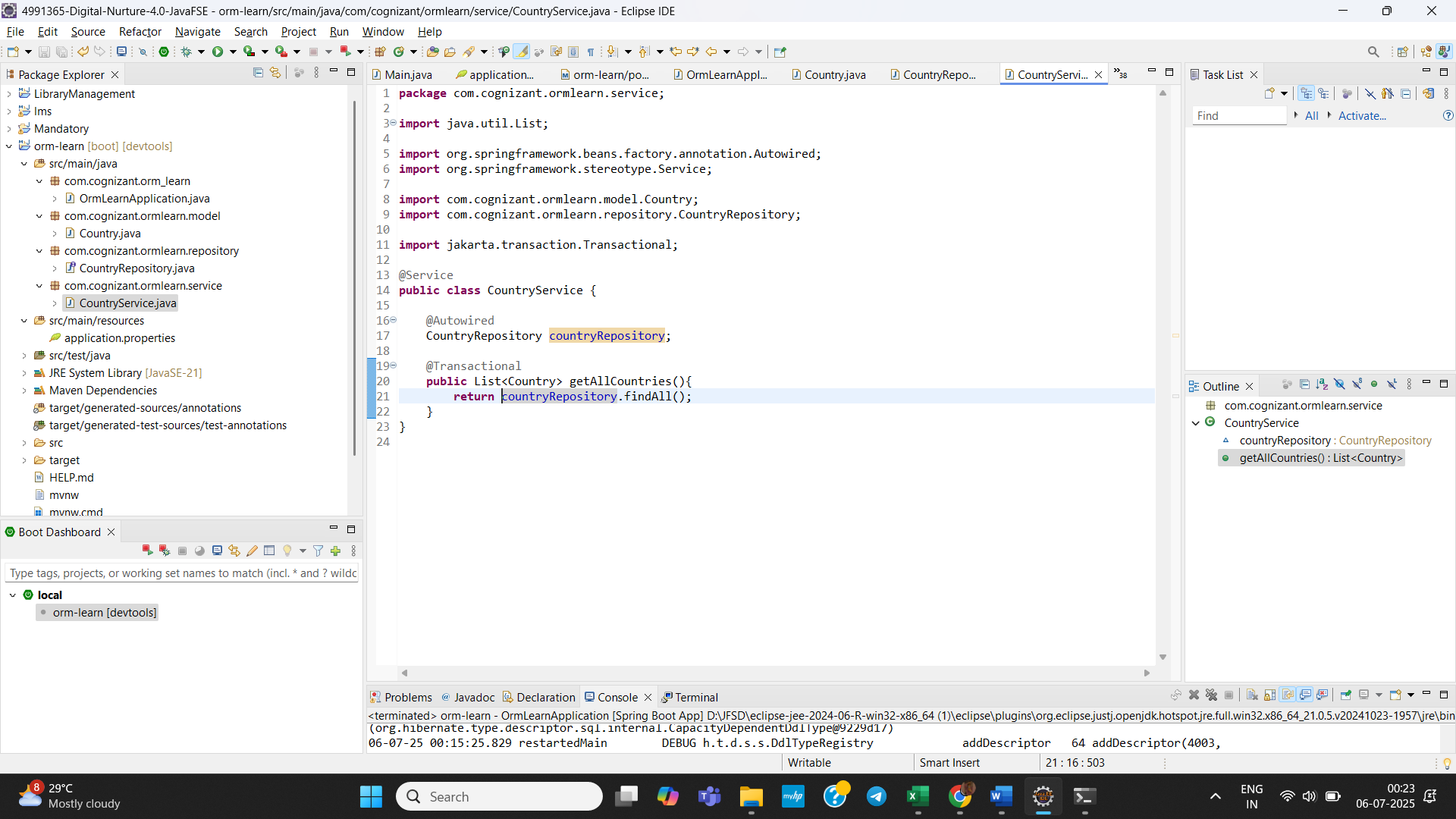
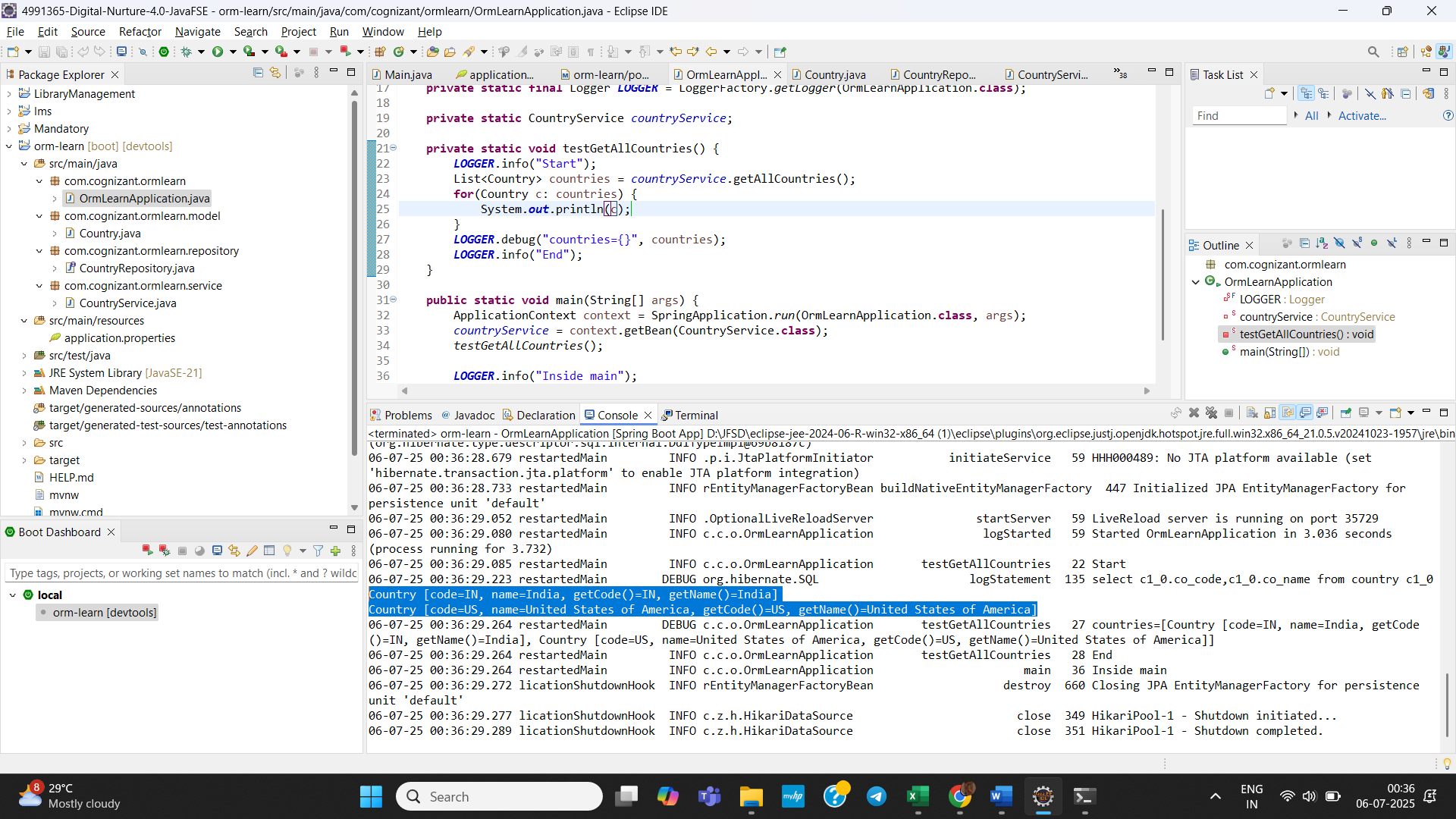
AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A computer screen with a black screen

AI-generated content may be incorrect.  A screenshot of a computer

AI-generated content may be incorrect.  A screenshot of a computer

AI-generated content may be incorrect.   

In Final Output, we retrieved the data from database.

1. **Difference between JPA, Hibernate and Spring Data JPA:**

**Java Persistence API (JPA):**

* A *specification* (interface-only) for object-relational mapping (ORM) in Java.
* Contains only interfaces, annotations like @Entity, @Id, @OneToMany, etc.
* Does not include an implementation – it relies on providers like Hibernate or EclipseLink.

**Hibernate :**

* A *concrete implementation* of the JPA specification.
* Also includes its own APIs, like Session, Criteria, HQL (Hibernate Query Language).

**Implementation :**

public Integer addEmployee(Employee employee) {

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

**Spring Data JPA :**

* A *Spring-based abstraction* on top of JPA providers (like Hibernate).
* Does not implement JPA but relies on providers like Hibernate.
* Removes boilerplate code.

**EmployeeRepository.java:**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java:**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}