EPAM COE JAVA

EPAM

A Training Report

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

Information and Technology

(COE JAVA)

LOVELY PROFESSIONAL UNIVERSITY PHAGWARA, PUNJAB

Submitted by

Name of the student **– K. Yovaraj Singh**

Registration number – **11901230**

Name of the supervisor – **Komal Arora**

Designation – **Assistant Professor**

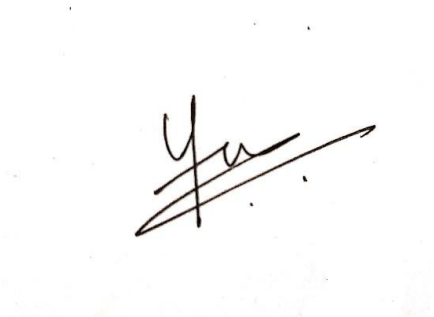
To whom it may concern,

I, K.Yovaraj Singh , 11901230, hereby declare that the work done by me on"COE JAVA" from 13th January-2023 to 28th April 2023, under the Internal supervision of Komal Arora from Lovely Professional University, Phagwara, Punjab, is a record of original work for the partial fulfilment of the requirements for the award of the degree Computer Science and Engineering.

Name of the Student (Registration Number)

K.Yovaraj Singh (l 1901230)

K. Yovaraj Singh,



Signature of the student :

Undertaking by the student for submitting Final Certificate of four months/one year Internship/OJT

Reg No. 11901230 Student Name: K.Yovaraj Singh

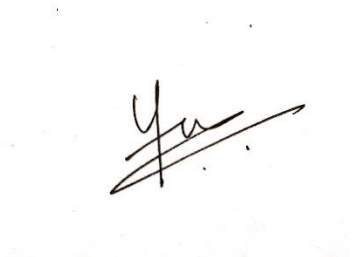
Program Name: B Tech, IT Batch Year: 2019-2023

Course Code: P133 Mobile No: 7005710831

I understand that I have been provisionally allowed to appear for the ETP viva and I hereby shall submit my final certificate of 4 months Internship/OJT to university after completion of my Internship/OJT but not later than May 2023.

I am aware that in case, I am unable to submit the same till the above-mentioned date, my final evaluation of internship/OJT shall be discarded by the university, and I grade shall be awarded in the result.

Signature of Student Signature of TPC-School Signature of HOS



Acknowledegment

I am incredibly grateful to my internship supervisor Komal Arora Ma’am for their invaluable guidance and support throughout my internship. From the moment I started, this training it took the time to get to know me and understand my goals for the internship. They provided me with clear direction and expectations, and were always available to answer my questions and provide valuable feedback.

Throughout the internship ,Komal Arora Ma’am provided me with invaluable insights and advice that helped me to grow as a professional. Their constructive feedback helped me to improve my skills and approach to my tasks, and their encouragement kept me motivated and focused. I am deeply thankful for Komal Arora Ma’am for the time and effort, and for their commitment to my success.

Table of Content

Unit 1 – Version Control Git , Git-Lab

Introduction:-

Git is a version control system (VCS) that is used to manage source code and track changes made to it. It was created by Linus Torvalds in 2005 and has since become one of the most popular VCS tools used by software developers worldwide. Git allows multiple developers to work on the same codebase simultaneously without interfering with each other's work. It also enables developers to collaborate on code development projects, review changes, and roll back changes when necessary.

Version Control Concept and its types:-

Version control is the practice of managing changes to documents, code, or any other type of file over time. It allows multiple people to collaborate on a project and keep track of the changes made to the project's files.

In the context of software development, version control is essential because it enables developers to keep track of changes to code over time. Version control systems (VCS) such as Git, SVN, and Mercurial store a complete history of changes to code files, making it easy to revert to previous versions if necessary. Version control also allows developers to collaborate on code development projects, working on different parts of the codebase simultaneously without interfering with each other's work.

Version control systems use a set of concepts and terminology to manage changes to files. Some of the key concepts include:

1.Repository: A repository is a central storage location for files managed by a version control system. It contains all versions of the files as well as information about who made changes and when.

2.Commit: A commit is a snapshot of changes to one or more files. When a developer makes changes to a file, they can create a commit to save those changes to the repository. Each commit is identified by a unique identifier, such as a hash value.

3.Branch: A branch is a separate line of development within a repository. Developers can create branches to work on new features or bug fixes without affecting the main development line. Once the changes in a branch are complete, they can be merged back into the main development line.

4.Merge: Merging is the process of combining changes from one branch into another. When two branches contain changes to the same files, merging allows developers to combine those changes into a single version of the file.

Why should we use Git:-

There are several reasons why a development team might choose to use GitLab:

1.Centralized code repository: GitLab provides a centralized location for storing code and other project-related files. This makes it easy for team members to access and collaborate on the same codebase.

2.Powerful collaboration features: GitLab has a robust set of collaboration features, such as merge requests, code reviews, and inline commenting, which make it easier for team members to work together on code changes.

3.Continuous Integration/Continuous Deployment (CI/CD): GitLab has built-in CI/CD tools that allow teams to automate the testing, building, and deployment of their code. This saves time and reduces the risk of errors that can occur during manual processes.

4.Issue tracking: GitLab has a built-in issue tracking system that allows teams to track bugs, feature requests, and other issues related to their codebase. This makes it easy to prioritize tasks and ensure that important issues are addressed in a timely manner.

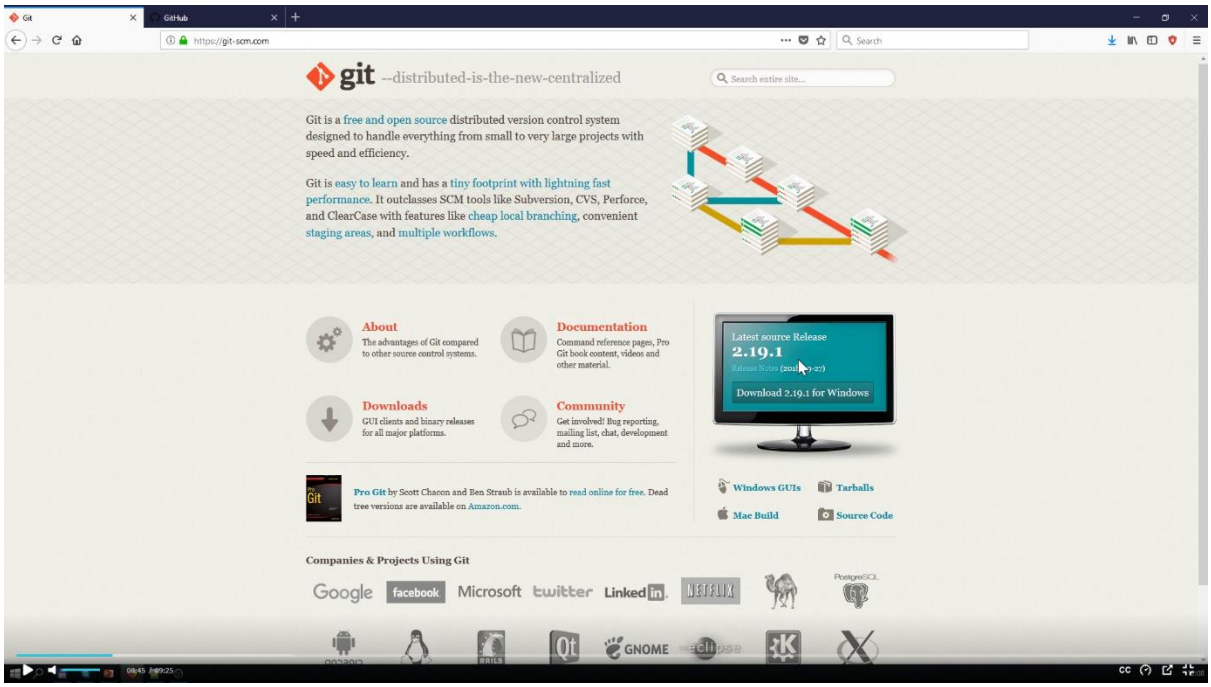
5.Security: GitLab has a strong focus on security and includes features such as two-factor authentication, role-based access control, and vulnerability scanning to help keep projects secure.

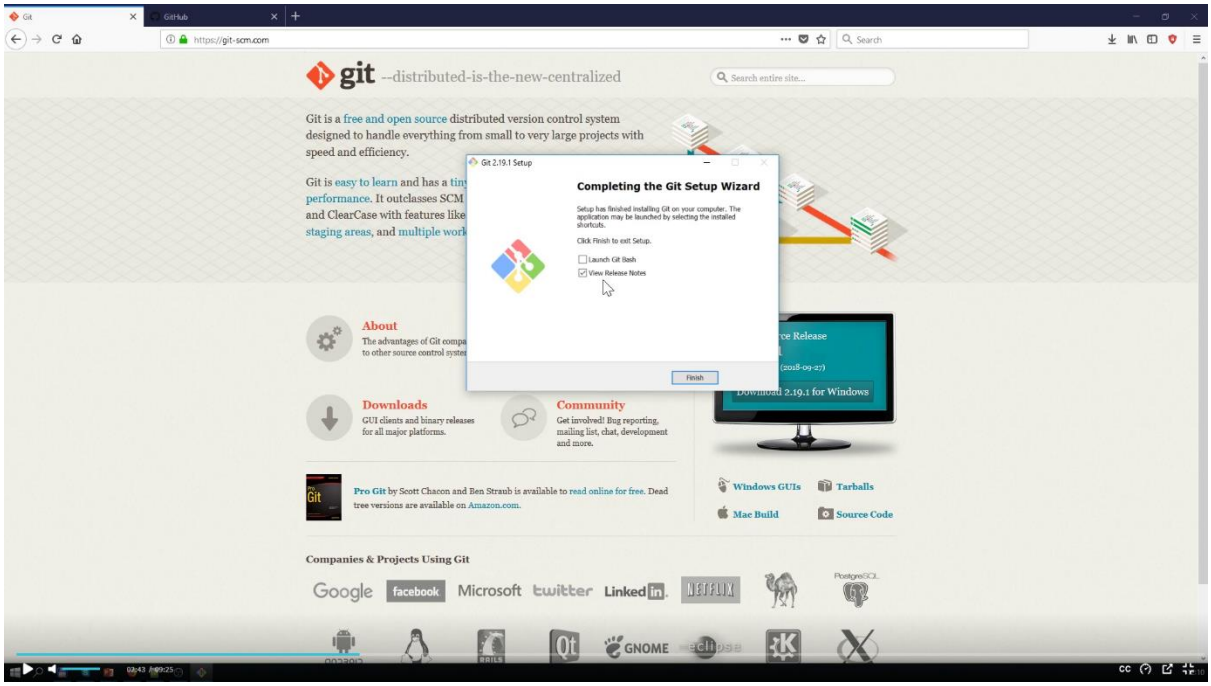
6.Flexibility: GitLab is a highly customizable platform that can be tailored to fit the needs of different development teams. It can be used with a variety of programming languages and integrates with many other tools and services.

Download Install and configure git:-

Step 1:-

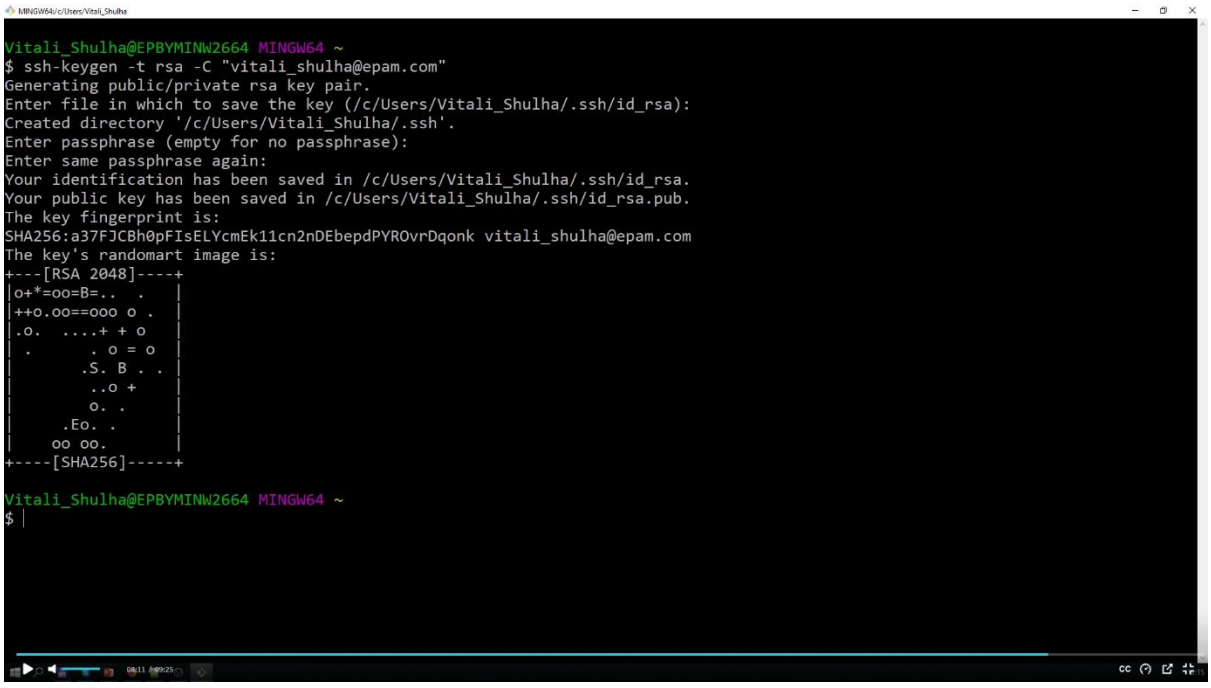
First you need to visit the site https://git-scm.com/ then click on the link download for windows after download kindly install all the process needed

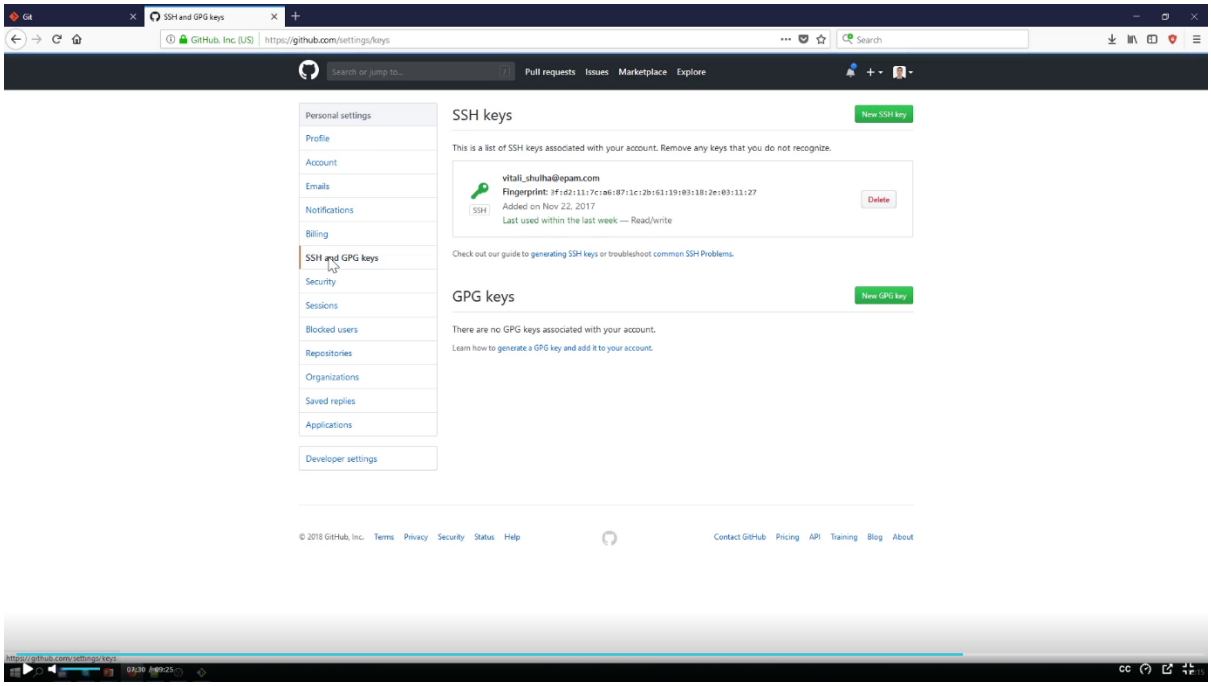




Step 2:-

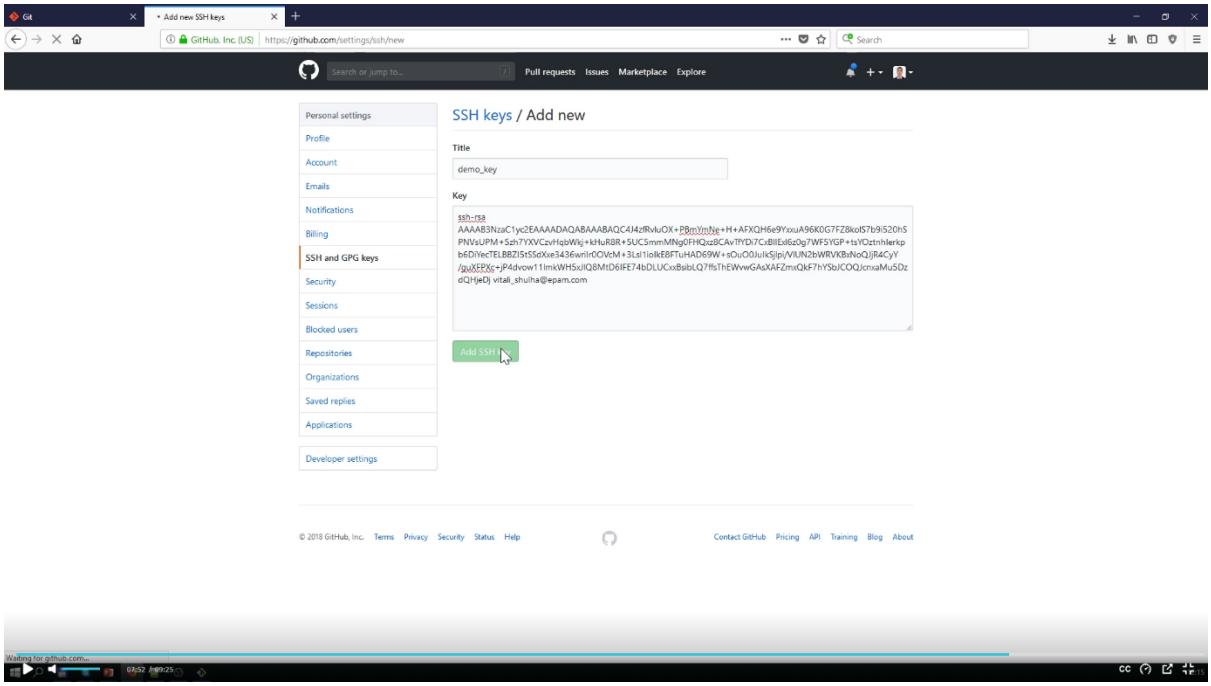
After installation try running by searching gitbash and we will be connecting with github with the use of SSH keys and store in a directory we will be generating the SSH keys from github as shown





Step 3:-

After that try pasting the private key in the SSH keys section and add them



Java Basics and DSA(Data Structures and Algorithm)

Introduction:-

The Java language was developed by a team of engineers from Sun Microsystems in 1995. However, it still maintains its top positions in the world rankings. Java can be used for development of any operating system. It is convenient and easy to master. This is a stable, reliable, and secure language.

Anyone can become a Java engineer who is prepared to learn continuously, to master new industrial trends, to proactively solve non-typical problems.

Where should you start your journey into the world of Java? First, from your willingness to learn. And second, by choosing a competently built training program.

The "Java Basics" course is the foundation that will allow you to immerse yourself in the Java world smoothly, as well as to perfect your knowledge and practical skills further on. In this course, you will explore the fundamental basics of the language and concepts of object-oriented programming, try launching your first application, practice using principal programming elements, create classes in Java, work with strings, and much more.

This is a self-study course. You can study according to your own schedule at a comfortable pace and return to the materials you have already studied, if necessary. A combination of theory using different formats (55%), practical exercises (40%), and self-check tests (5%) will help you better understand and remember what you study.

The ability to program in Java will open multiple job opportunities for you as part of interesting projects, give you excellent growth perspectives, boost your chances in the job market, and provide you with a fair and growing income.

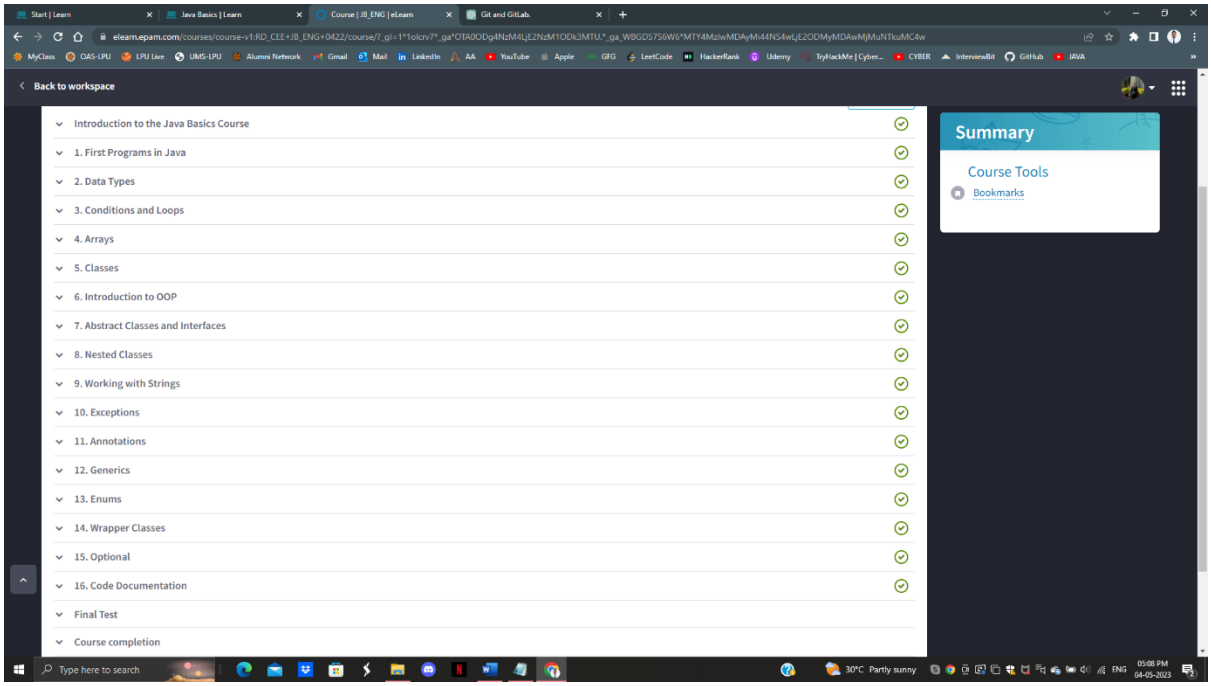
Course Goal:-

After taking the "Java Basics" course, you will be able to:

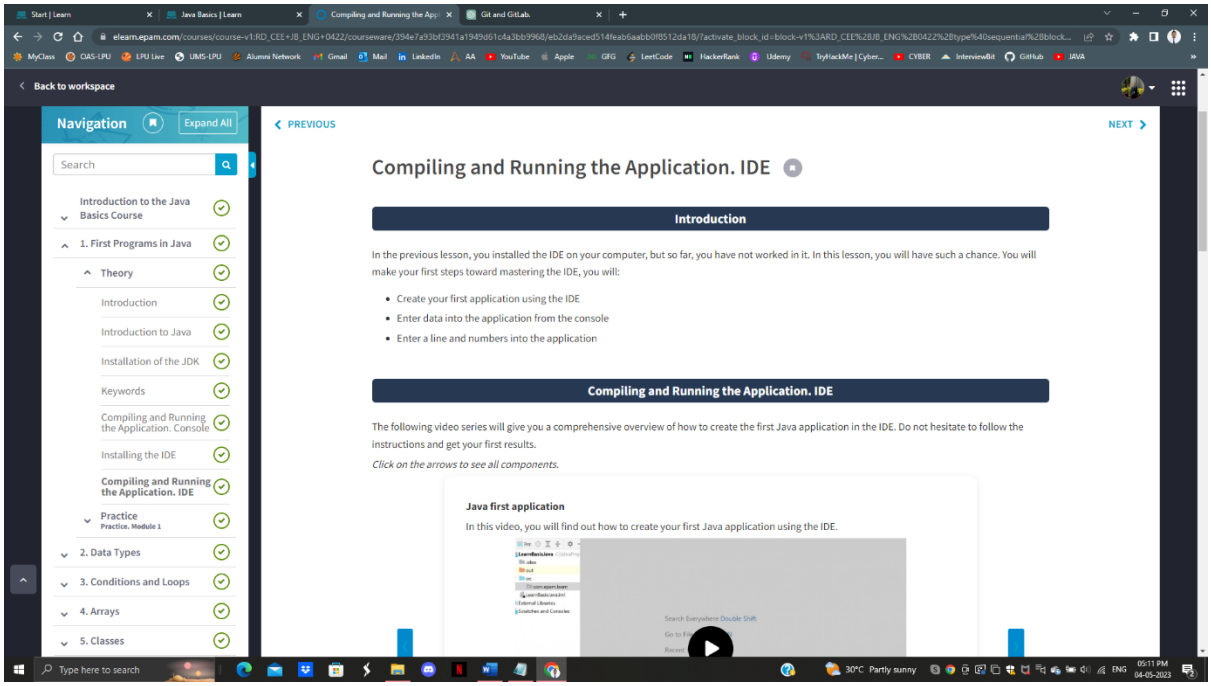
- define and explain the fundamental basics of the Java language and concepts of object-oriented programming;
- use principal programming elements, such as variables, statements, conditional statements, loops, methods;
- use different data types: primitives, objects, arrays, strings, enumerations;
- create classes in Java with a correct description of fields and methods;

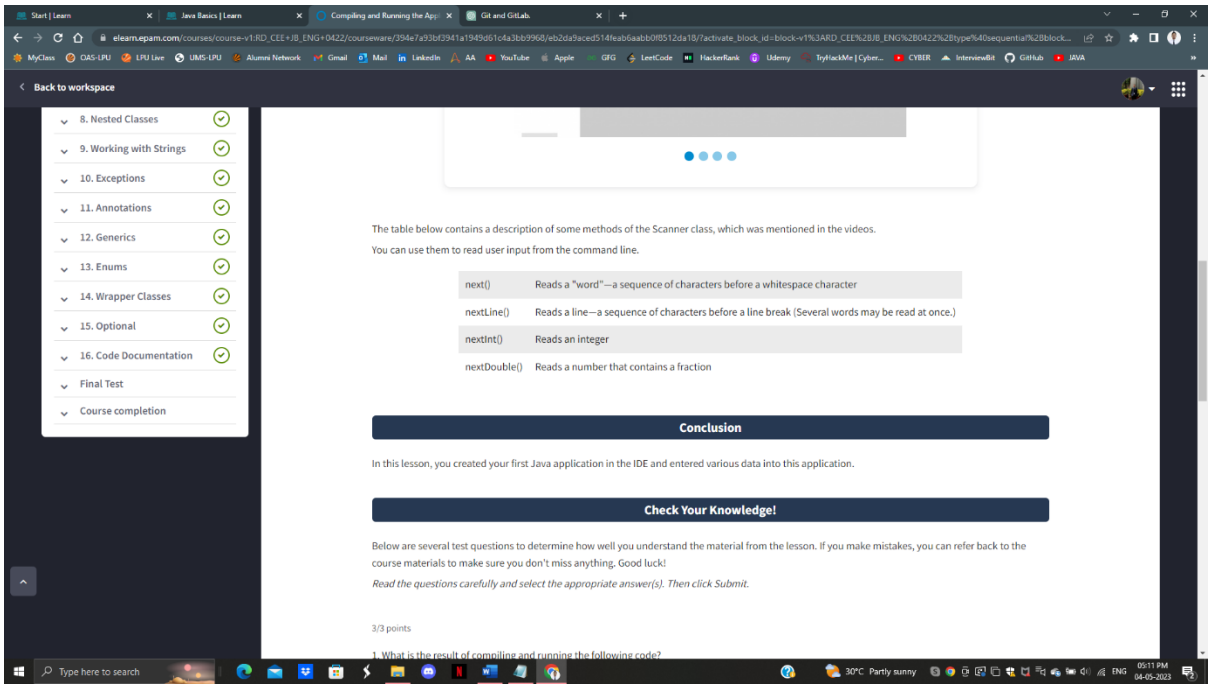
- apply the principles of encapsulation, inheritance, and polymorphism in practice;
- use the class Object and the methods contained therein;
- work with strings in Java using the class String and its methods, classes StringBuilder and StringBuffer;
- correctly work with exceptions, as well as create your own exceptions;
- correctly document your code using Javadoc;
- write your first programs in Java.

Screenshot of the courses under this program:-

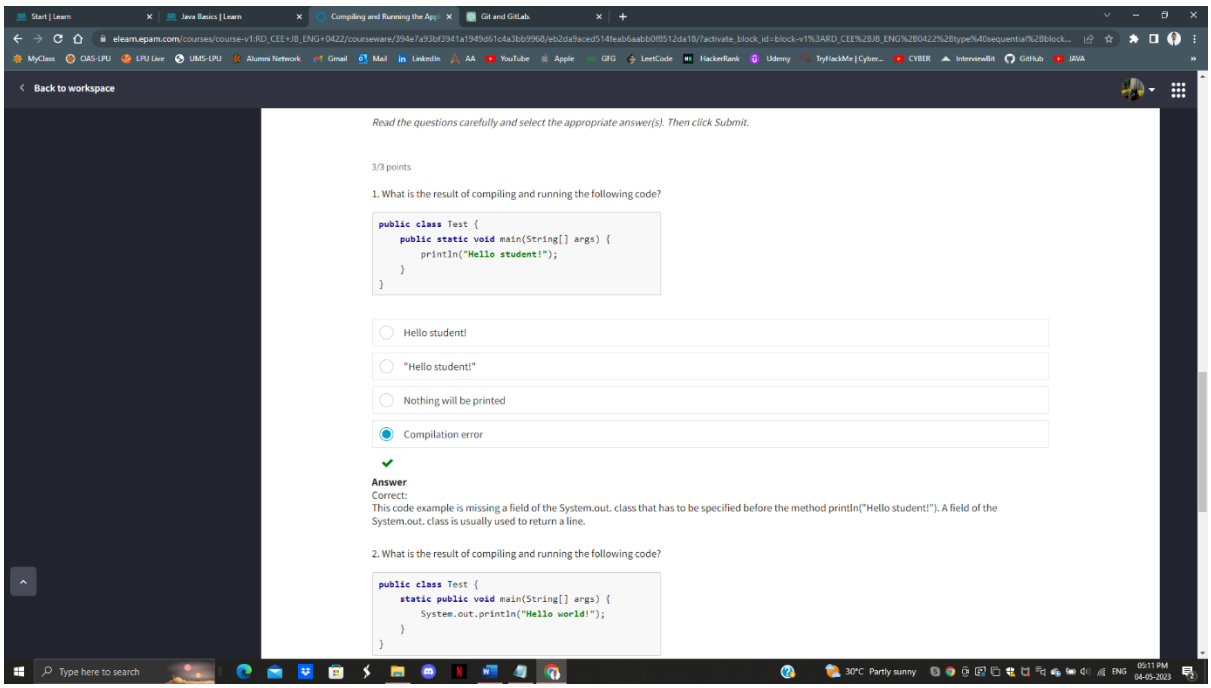


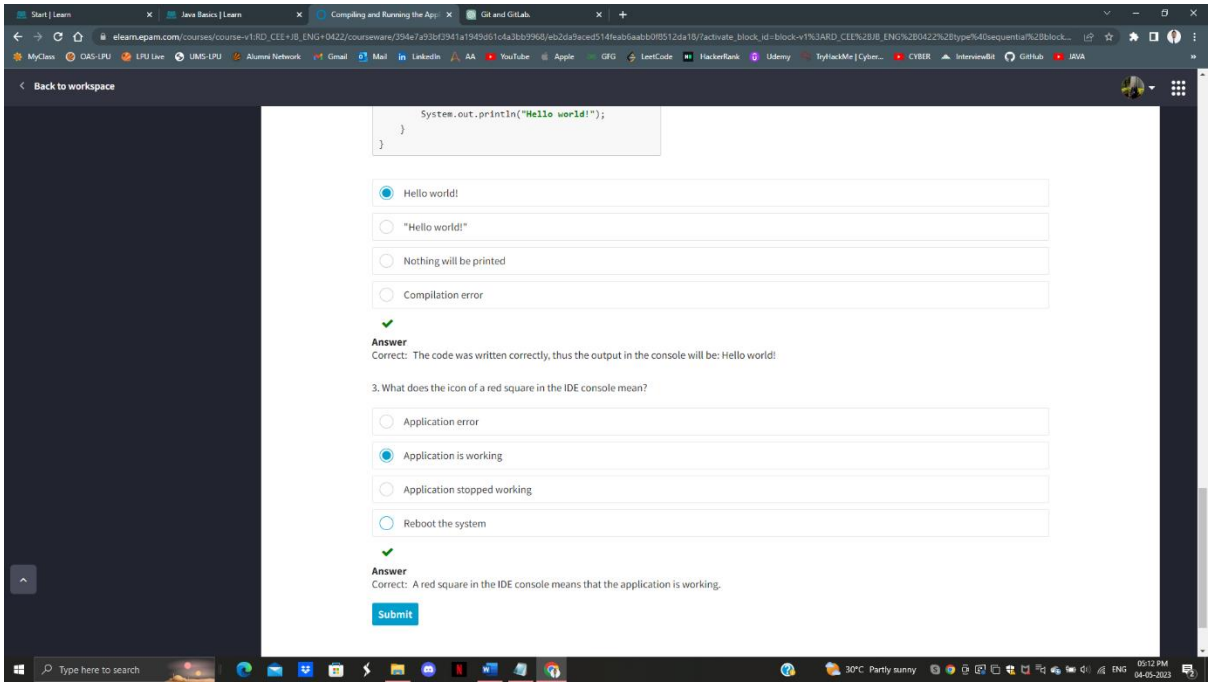
Talking about the first topic i.e we will be learning about the java programming language how to start and compile


Back to workspace
Navigation
Expand All
Search
Introduction to the Java Basics Course
1. First Programs in Java
Theory
Introduction
Introduction to Java
Installation of the JDK
Keywords
Compiling and Running the Application. Console
Installing the IDE
Compiling and Running the Application. IDE
Practice
Practice. Module 1
2. Data Types
3. Conditions and Loops
4. Arrays
5. Classes
PREVIOUS
NEXT
Compiling and Running the Application. IDE
Introduction
In the previous lesson, you installed the IDE on your computer, but so far, you have not worked in it. In this lesson, you will have such a chance. You will make your first steps toward mastering the IDE, you will:
Create your first application using the IDE
Enter data into the application from the console
Enter a line and numbers into the application
Compiling and Running the Application. IDE
The following video series will give you a comprehensive overview of how to create the first Java application in the IDE. Do not hesitate to follow the instructions and get your first results.
Click on the arrows to see all components.
Java first application
In this video, you will find out how to create your first Java application using the IDE.


Back to workspace
8. Nested Classes
9. Working with Strings
10. Exceptions
11. Annotations
12. Generics
13. Enums
14. Wrapper Classes
15. Optional
16. Code Documentation
Final Test
Course completion
The table below contains a description of some methods of the Scanner class, which was mentioned in the videos.
You can use them to read user input from the command line.
next() Reads a "word"—a sequence of characters before a whitespace character
nextLine() Reads a line—a sequence of characters before a line break (Several words may be read at once.)
nextInt() Reads an integer
nextDouble() Reads a number that contains a fraction
Conclusion
In this lesson, you created your first Java application in the IDE and entered various data into this application.
Check Your Knowledge!
Below are several test questions to determine how well you understand the material from the lesson. If you make mistakes, you can refer back to the course materials to make sure you don't miss anything. Good luck!
Read the questions carefully and select the appropriate answer(s). Then click Submit.
3/3 points

Also we will be getting some of the questions to solve to get a revision of the course we learned as show in the picture below





We have also leaned some of the topics of DSA(Data Structures and Algorithm) like Arrays etc,

What is an Array?

An **array** is a set of homogeneous data with a common name. Arrays help to group related data and may be used for different purposes. For example, in an array you can store data about monthly precipitation levels or the titles of books in your library. Arrays may be one-dimensional or multidimensional (but one-dimensional arrays are used more often).

The **key benefit of an array** is the possibility of quick access to its elements, which makes it easy and convenient to process data. For example, if you have an array of data about a student's marks in a specific subject, you can easily calculate the mean result. To do this, you need to iterate through the array elements in loops.

To create an array, you need to:

1. Declare an array
2. Allocate memory for the array
3. Initialize the array

Declare an array:-

method: <data type>[] <identifier>;

Allocating memory for an array:-

array = **new** int[5];

Initializing an array:-

int[] array = **new** int[3];

A simple program of an array:-

```
public class SimpleArrayExample {
    public static void main(String[] args) {
        // create an empty array with size 3
        int[] myArray = new int[3];

        // add elements to the array
        myArray[0] = 1;
        myArray[1] = 2;
        myArray[2] = 3;

        // print the contents of the array
        for (int i = 0; i < myArray.length; i++) {
            System.out.println(myArray[i]);
        }
    }
}
```

Output:-

```
1
2
3
```

While learning about this course we have also got to know about the uses of OOPS in java which is also known as object oriented programming in java

Introduction:-

In Java, OOPs is a core concept, and almost everything in Java is an object. Java supports the four pillars of OOPs, which are:

Encapsulation: This means that the state (variables) and behavior (methods) of an object are bundled together and kept private, so that they can be accessed and modified only through well-defined interfaces. Encapsulation helps to hide the complexity of an object and make it easier to use.

Program on Encapsulation:-

```
public class test Encapsulation
{
public static void main(String args[])
{
Encapsulation obj = new Encapsulation();
obj.setName(“harsh”);
obj.setAge(19);
obj.setRoll(51);

System.out.print(“Geek’s name:”+ obj.getName());
System.out.print(“Geek’s age:”+ obj.getAge());
System.out.print(“Geek’s roll:”+ obj.getRoll());

}
}
```

Inheritance: This is a mechanism by which a new class can be derived from an existing class. The new class (the subclass) inherits all the properties and methods of the existing class (the superclass) and can also add new properties and methods of its own. Inheritance allows for code reuse and can make the code more organized and easier to maintain.

Program for Inheritance:-

```
class Employee
{
int salary = 4000;
}
class Programmer extends Employee
{
int bonus = 1000;
public static void main(String args[])
{
programmer p = new programmer ();
System. out. print("Programmer. salary is :"+p . salary)
System. out. print("Bonus of Programmer is :"+p . bonus)
}
}
```

Polymorphism: This means the ability of an object to take on many forms. In Java, polymorphism can be achieved through method overriding and method overloading. Method overriding allows a subclass to provide its own implementation of a method that is already defined in the superclass. Method

overloading allows multiple methods with the same name but different parameters to coexist in a class.

Program for Polymorphism:-

```
class Addition
{
static int sum (int a , int b)
{
return (a+b);
}
static double sum(int a, int b)
{
return (a+b);
}
}
class Test
{
public static void main(String args[])
{
System.out.println(Addition.sum(5,5));
}
}
```

Abstraction: This refers to the process of hiding the implementation details of an object and only exposing the essential features to the user. Abstraction is achieved in Java through abstract classes and interfaces. An abstract class is a class that cannot be instantiated and is intended to be subclassed by concrete classes that provide the actual implementation. An interface is a collection of

abstract methods that can be implemented by any class that implements the interface.

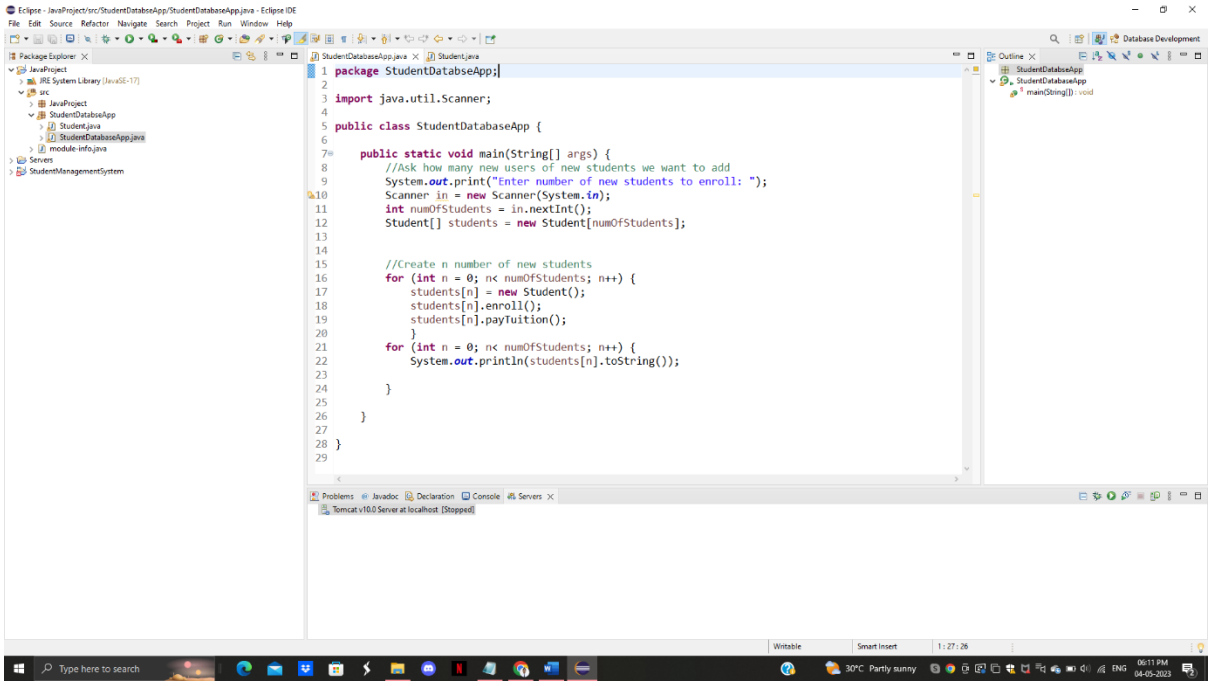
Program for Abstract :-

```
abstract class Animal
{
public abstract void sound();
}
class Dog extends Animal
{
public void sound()
{
System.out.println(“Woof”);
}
public static void main(String args[])
{
Animal.obj = new Dog;
obj.sound();
}
}
```

Also talking about the course we have also made a project based on the topic of java

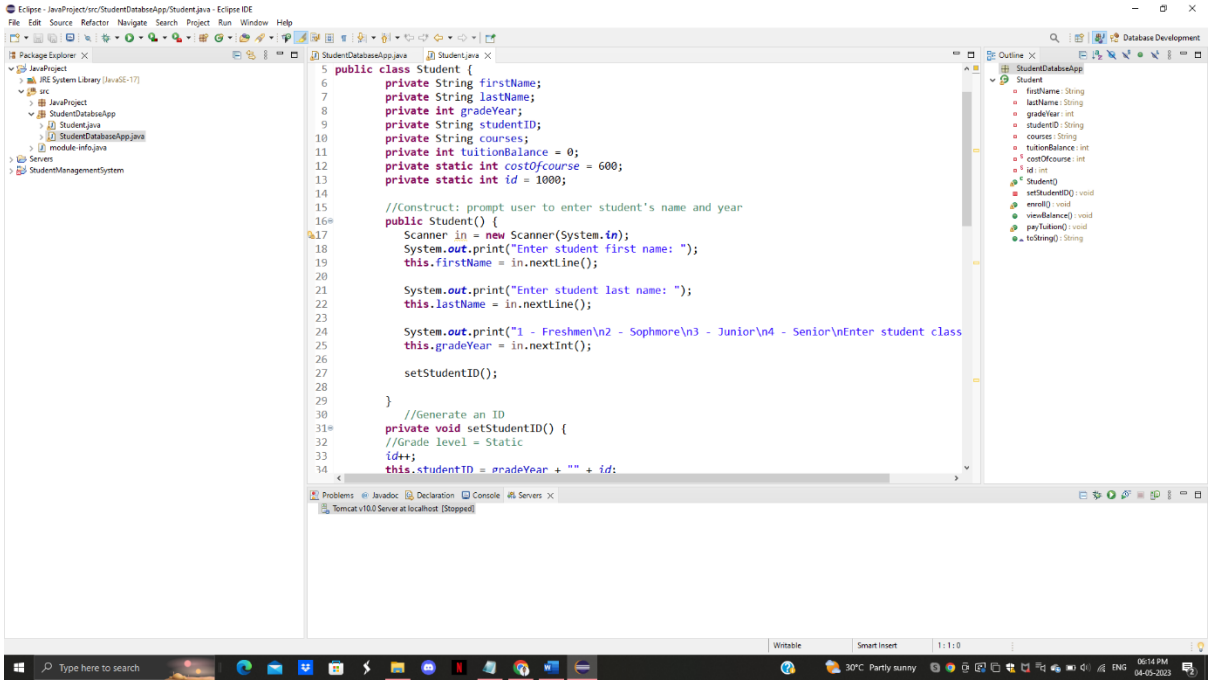
Project name – Student Management System:-

The picture below shows the implementation of a class in which the user will be asking about the number of students we want to and and create the students .

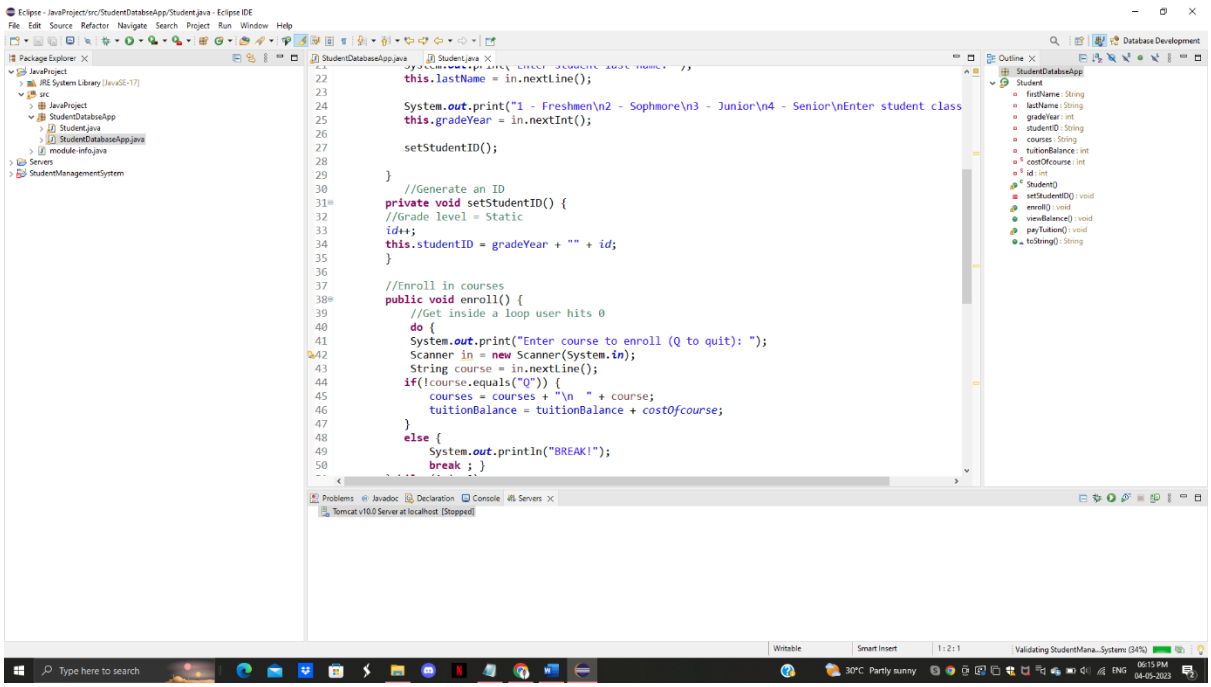


```
package StudentDatabseApp;

import java.util.Scanner;

public class StudentDatabaseApp {

    public static void main(String[] args) {
        //Ask how many new users of new students we want to add
        System.out.print("Enter number of new students to enroll: ");
        Scanner in = new Scanner(System.in);
        int numOfStudents = in.nextInt();
        Student[] students = new Student[numOfStudents];


        //Create n number of new students
        for (int n = 0; n< numOfStudents; n++) {
            students[n] = new Student();
            students[n].enroll();
            students[n].payTuition();
            }
        for (int n = 0; n< numOfStudents; n++) {
            System.out.println(students[n].toString());

        }

    }
}
```

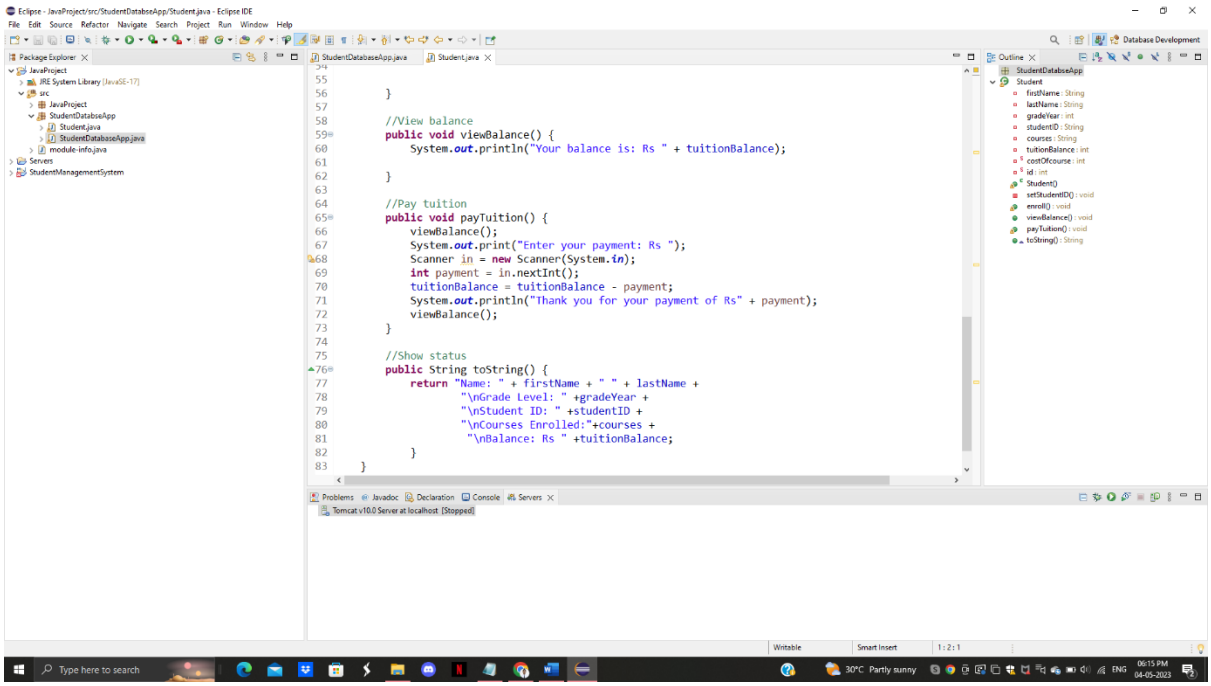
This the continuation of the first class in a second class in order to make the user enter the student's name and the year along with the generated ID and courses enrolled with a price tag to pay for it



```
public class Student {
        private String firstName;
        private String lastName;
        private int gradeYear;
        private String studentID;
        private String courses;
        private int tuitionBalance = 0;
        private static int costOfcourse = 600;
        private static int id = 1000;

        //Construct: prompt user to enter student's name and year
        public Student() {
           Scanner in = new Scanner(System.in);
           System.out.print("Enter student first name: ");
           this.firstName = in.nextLine();

           System.out.print("Enter student last name: ");
           this.lastName = in.nextLine();

           System.out.print("1 - Freshmen\n2 - Sophmore\n3 - Junior\n4 - Senior\nEnter student class
           this.gradeYear = in.nextInt();

           setStudentID();

        }
           //Generate an ID
        private void setStudentID() {
        //Grade level = Static
        id++;
        this.studentID = gradeYear + "" + id;
```

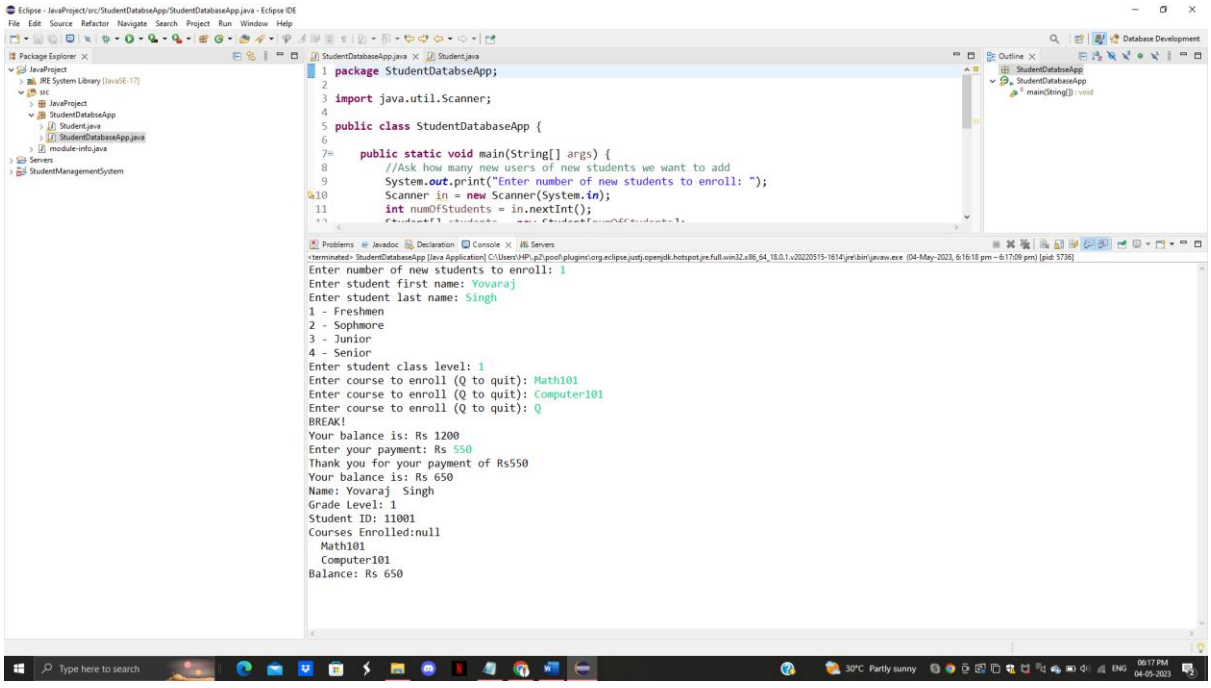


```
           this.lastName = in.nextLine();

           System.out.print("1 - Freshmen\n2 - Sophmore\n3 - Junior\n4 - Senior\nEnter student class
           this.gradeYear = in.nextInt();

           setStudentID();

        }
           //Generate an ID
        private void setStudentID() {
        //Grade level = Static
        id++;
        this.studentID = gradeYear + "" + id;
        }

        //Enroll in courses
        public void enroll() {
            //Get inside a loop user hits 0
            do {
            System.out.print("Enter course to enroll (Q to quit): ");
            Scanner in = new Scanner(System.in);
            String course = in.nextLine();
            if(!course.equals("Q")) {
                courses = courses + "\n  " + course;
                tuitionBalance = tuitionBalance + costOfcourse;
            }
            else {
                System.out.println("BREAK!");
                break ; }
```



```
        }

        //View balance
        public void viewBalance() {
            System.out.println("Your balance is: Rs " + tuitionBalance);

        }

        //Pay tuition
        public void payTuition() {
            viewBalance();
            System.out.print("Enter your payment: Rs ");
            Scanner in = new Scanner(System.in);
            int payment = in.nextInt();
            tuitionBalance = tuitionBalance - payment;
            System.out.println("Thank you for your payment of Rs" + payment);
            viewBalance();
        }

        //Show status
        public String toString() {
            return "Name: " + firstName + " " + lastName +
                    "\nGrade Level: " +gradeYear +
                    "\nStudent ID: " +studentID +
                    "\nCourses Enrolled:"+courses +
                     "\nBalance: Rs " +tuitionBalance;
            }
    }
```

The picture below contains the screenshot of the output :-



```
package StudentDatabseApp;

import java.util.Scanner;

public class StudentDatabaseApp {

    public static void main(String[] args) {
        //Ask how many new users of new students we want to add
        System.out.print("Enter number of new students to enroll: ");
        Scanner in = new Scanner(System.in);
        int numOfStudents = in.nextInt();
```

```
Enter number of new students to enroll: 1
Enter student first name: Yovaraj
Enter student last name: Singh
1 - Freshmen
2 - Sophmore
3 - Junior
4 - Senior
Enter student class level: 1
Enter course to enroll (Q to quit): Math101
Enter course to enroll (Q to quit): Computer101
Enter course to enroll (Q to quit): Q
BREAK!
Your balance is: Rs 1200
Enter your payment: Rs 550
Thank you for your payment of Rs550
Your balance is: Rs 650
Name: Yovaraj  Singh
Grade Level: 1
Student ID: 11001
Courses Enrolled:null
  Math101
  Computer101
Balance: Rs 650
```

Java Web Development with Spring Core

Spring makes building web applications fast and hassle-free. By removing much of the boilerplate code and configuration associated with web development, you get a modern web programming model that streamlines the development of server-side HTML applications, REST APIs, and bidirectional, event-based systems.

Spring Boot is the starting point of your developer experience, whatever you're building. Spring Boot is designed to get you up and running as quickly as possible, with minimal upfront configuration. With its embedded application servers, you can be serving in seconds.

Spring's out-of-the-box, production-ready features (like tracing, metrics, and health status) provide developers with deep insight into their applications.

Finally, Spring supports multiple JVM languages: Java, Kotlin, and Groovy.

Spring MVC (Model-View-Controller) is a web framework based on the Spring Framework that provides a clean separation of concerns between the different layers of a web application.

In Spring MVC, the Model layer represents the data and business logic of the application, the View layer represents the user interface, and the Controller layer handles the user input and coordinates the interaction between the Model and View.

The key features of Spring MVC are:

1.Flexible and configurable: Spring MVC provides a flexible and configurable architecture that can be easily customized to suit the needs of different web applications.

2.Dependency Injection: Spring MVC is built on the Spring Framework, which provides powerful dependency injection capabilities that make it easy to manage the different components of a web application.

3.Request handling: Spring MVC provides a powerful request handling mechanism that can handle different types of requests, including GET, POST, PUT, DELETE, and more.

4.View resolution: Spring MVC provides a view resolution mechanism that can automatically select the appropriate view based on the request and the configuration of the application.

5.Validation: Spring MVC provides a built-in validation framework that can be used to validate user input and ensure that it meets the requirements of the application.

6.Testing: Spring MVC provides a testing framework that makes it easy to write unit tests for controllers and other components of the application.

Spring MVC provides several annotations that can be used to configure various aspects of a web application. Here are some of the most commonly used annotations:

1.@Controller: This annotation is used to mark a class as a controller. Controllers handle user requests and return the appropriate response.

2.@RequestMapping: This annotation is used to map a URL to a specific controller method. You can specify the HTTP method (GET, POST, etc.), the URL pattern, and any request parameters or headers.

3.@PathVariable: This annotation is used to extract a variable from the URL path and pass it as a method parameter. For example, if you have a URL pattern

/users/{id}, you can use **@PathVariable**("id") to extract the id variable from the URL.

4.@RequestParam: This annotation is used to extract a request parameter and pass it as a method parameter. For example, if you have a request parameter name, you can use @RequestParam("name") to extract the name parameter from the request.

5.@RequestBody: This annotation is used to extract the request body and convert it to a Java object. This is typically used for handling JSON or XML requests.

6.@ResponseBody: This annotation is used to convert a method's return value to a JSON or XML response. This is typically used for returning data from a REST API.

7.@ModelAttribute: This annotation is used to bind a method parameter to a model attribute. Model attributes are used to pass data between the controller and the view.

8.@SessionAttribute: This annotation is used to bind a method parameter to a session attribute. Session attributes are stored in the user's session and persist across multiple requests.

Java Web Development with Spring Boot

Spring Boot is an open-source Java-based framework that simplifies the development of web applications and microservices. It is built on top of the Spring Framework and provides a set of pre-configured features and conventions that make it easy to create standalone, production-grade Spring-based applications with minimal configuration.

Spring Boot includes a wide range of features, including embedded servers, auto-configuration, and dependency management, that help developers get up and running quickly. With Spring Boot, developers can focus on writing business logic instead of spending time on configuration and boilerplate code.

Some of the key benefits of using Spring Boot include faster development, improved productivity, and easier deployment. Spring Boot also provides a wide range of integrations with other technologies, such as databases, messaging systems, and security frameworks.

Overall, Spring Boot is a powerful framework that simplifies the development of Spring-based applications, making it an excellent choice for building modern, scalable, and robust web applications and microservices.

Why we use spring boot:-

There are several reasons why developers use Spring Boot for building web applications and microservices:

1.Simplified Configuration: Spring Boot provides a set of pre-configured defaults and auto-configuration, which makes it easy to get started with Spring-based applications. Developers can focus on writing business logic instead of spending time on configuration and boilerplate code.

2.Rapid Development: With Spring Boot, developers can quickly prototype and develop applications. It provides a wide range of features such as embedded servers, hot reloading, and a powerful command-line interface that speeds up the development process.

3.Extensive Ecosystem: Spring Boot has a large and active ecosystem, with a wide range of plugins, libraries, and integrations available. It integrates seamlessly with other Spring projects, as well as with popular technologies such as databases, messaging systems, and security frameworks.

4.Production-Ready: Spring Boot is designed to be production-ready, with features such as health checks, metrics, and logging built-in. It also provides support for monitoring and management, making it easier to deploy and manage applications in production.

5.Community Support: Spring Boot has a large and active community of developers, who contribute to the project, provide support, and share best practices. This community support makes it easier to learn and use Spring Boot effectively.

Annotations in spring boot:-

Spring Boot is built on top of the Spring Framework and provides a number of additional features and conventions to simplify the development of web applications. Here are some of the most commonly used annotations in Spring Boot:

@SpringBootApplication: This annotation is used to mark the main class of a Spring Boot application. It combines several other annotations **(@Configuration**, **@EnableAutoConfiguration**, and **@ComponentScan**) into a single annotation.

@RestController: This annotation is used to mark a class as a controller that handles RESTful requests. It is similar to the **@Controller** annotation in Spring MVC, but with the added benefit of automatically applying the **@ResponseBody** annotation to all methods.

@GetMapping, **@PostMapping**, **@PutMapping, @DeleteMapping**: These annotations are used to map HTTP requests to specific controller methods based on the HTTP method. For example, **@GetMapping** maps a GET request to a controller method.

@PathVariable: This annotation is used to extract a variable from the URL path and pass it as a method parameter.

@RequestParam: This annotation is used to extract a request parameter and pass it as a method parameter.

@RequestBody: This annotation is used to extract the request body and convert it to a Java object.

@Autowired: This annotation is used to inject a dependency into a class. Spring Boot will automatically create and manage instances of classes annotated with **@Component**, **@Service**, or **@Repository**, and inject them into other classes that have an **@Autowired** annotation.

@ConfigurationProperties: This annotation is used to map external configuration properties to a Java object. This can be used to configure application settings from an external properties file or environment variables.

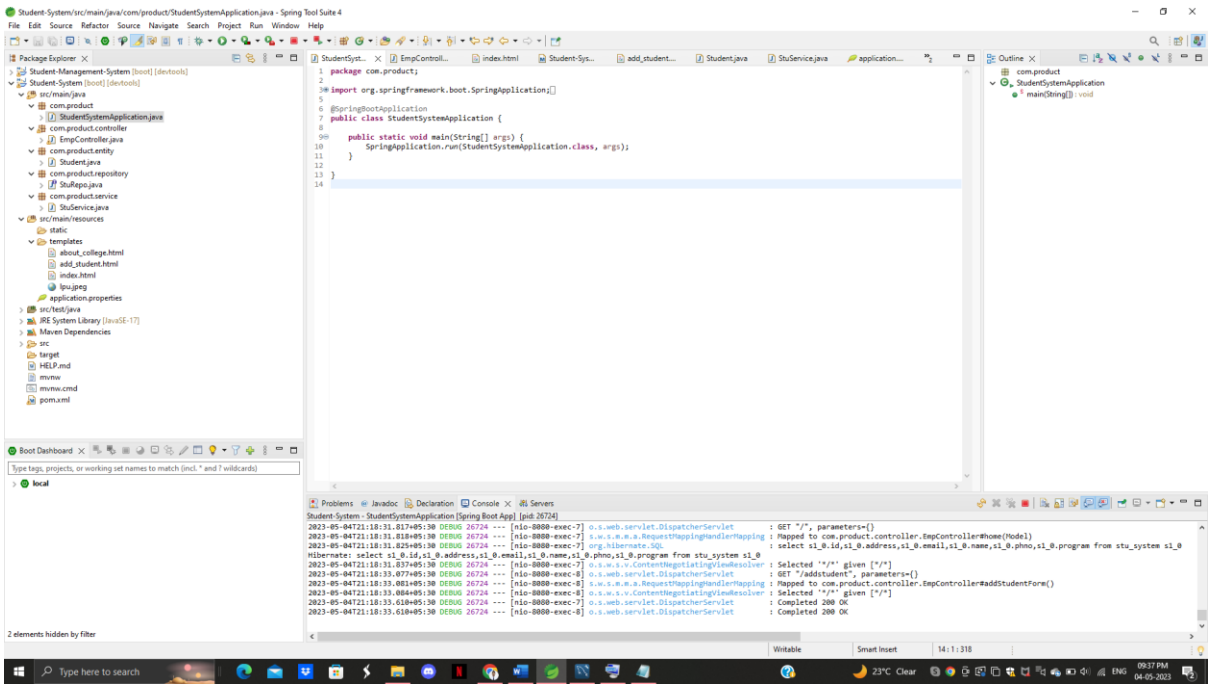
After the completion of this course we have been provided to make a project on a specific topic

Project name:- Student Management System

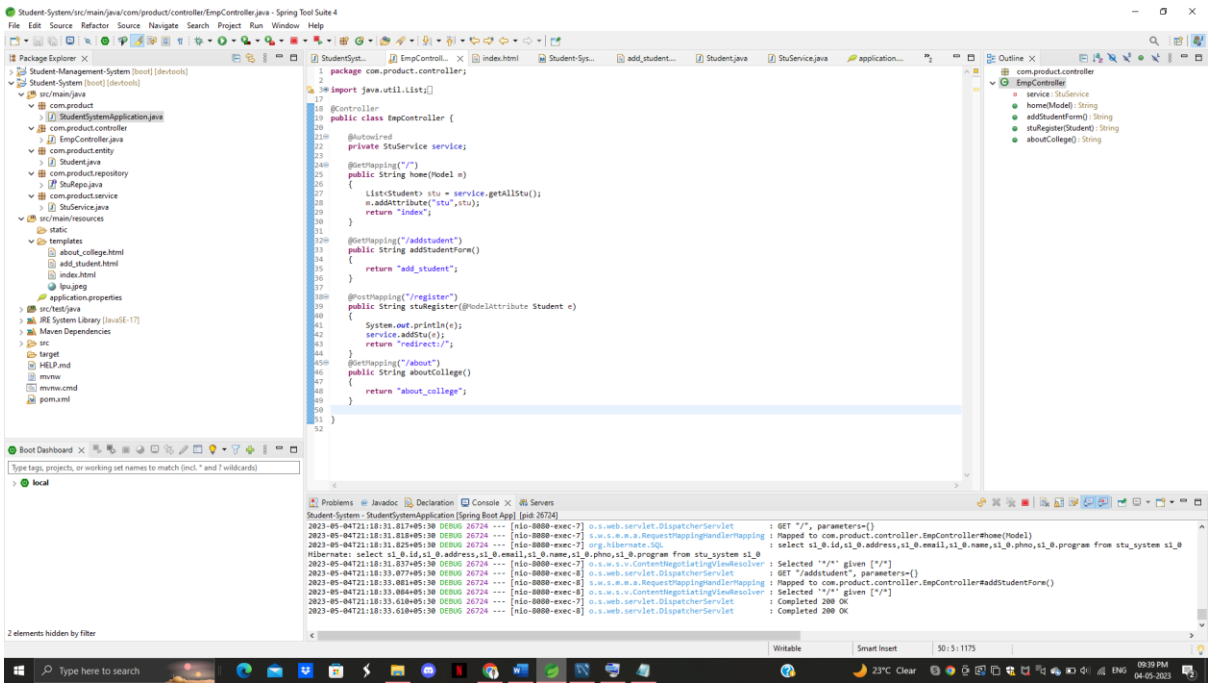
For making this project we have use Spring Tool Suite 4 as it is a spring boot application for making spring boot projects

In Spring Boot, the main class serves as the entry point for the application. It is responsible for configuring and starting the Spring Boot application.

The main class also typically contains the main method, which is the starting point for the application. This method calls the SpringApplication.run() method to start the Spring Boot application, which in turn starts the embedded Tomcat server and initializes the Spring ApplicationContext.

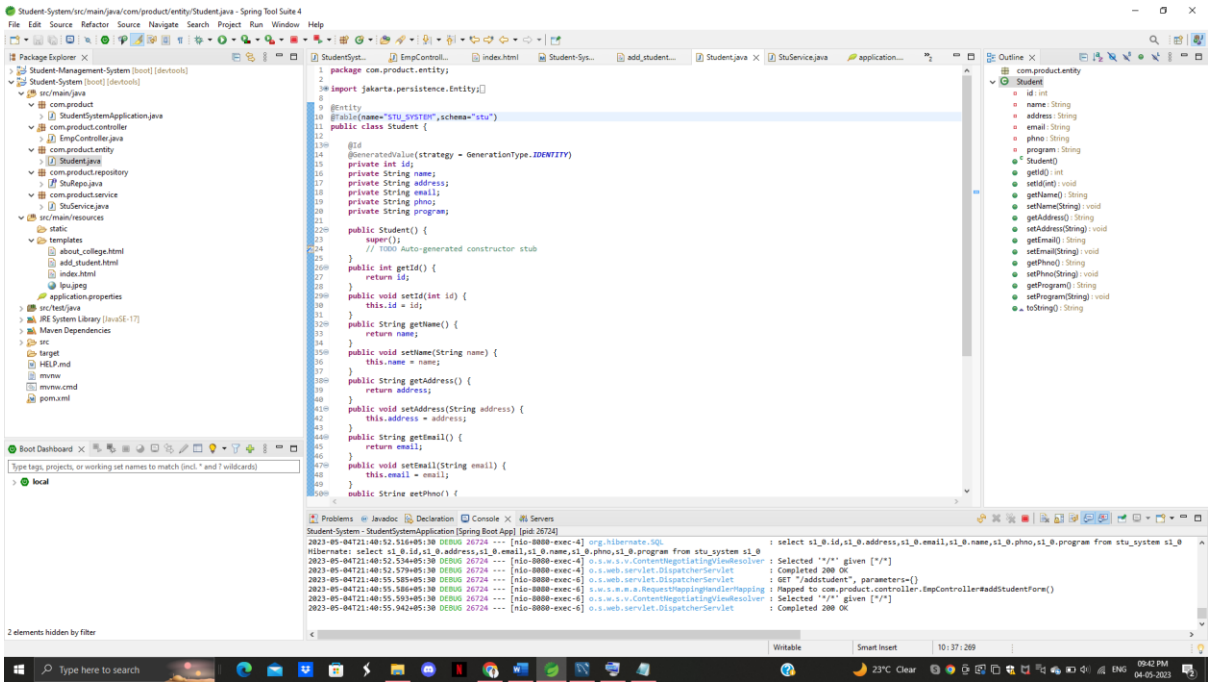


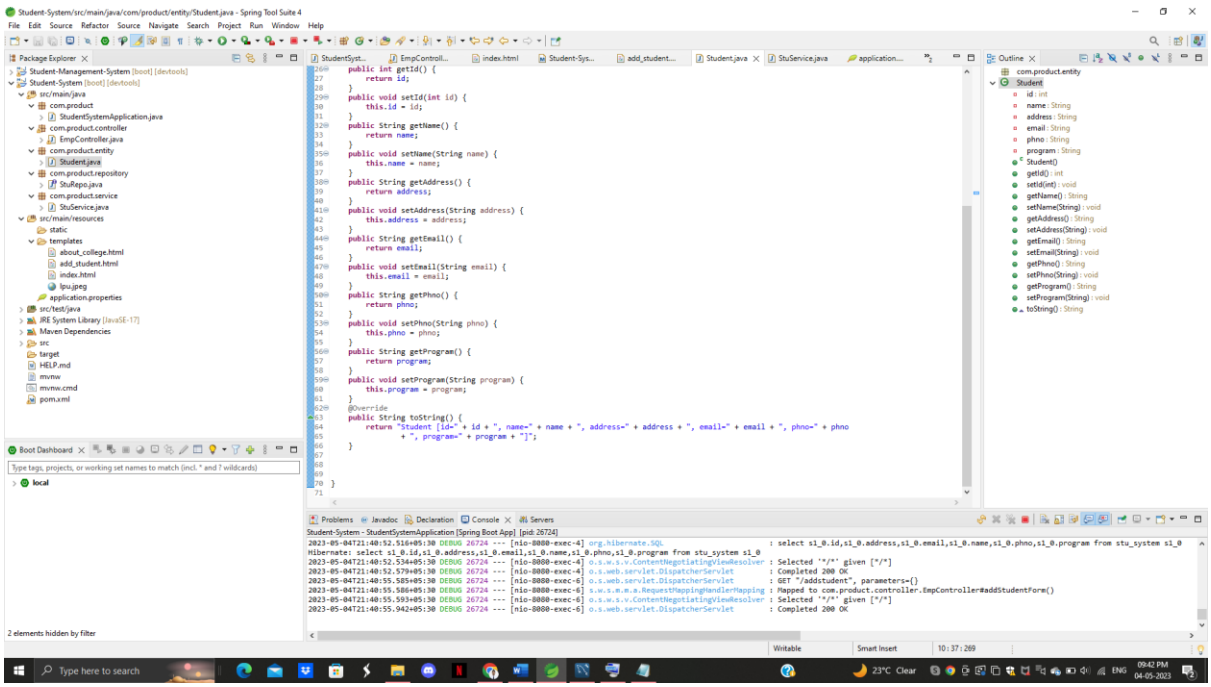
In Spring Boot, the Controller class is responsible for handling incoming HTTP requests and returning the appropriate HTTP responses. Controller classes contain methods that are annotated with various HTTP method annotations like **@GetMapping**, **@PostMapping, @PutMapping**, **@DeleteMapping**, etc. These annotations map the HTTP request methods to the corresponding methods in the controller class.



In Spring Boot, an Entity class is a Java class that represents a table in a relational database. An Entity class is typically annotated with **@Entity** to indicate that it is a persistent class and should be mapped to a database table.

Entity classes usually contain fields that represent the columns in the database table, and methods to access and manipulate those fields. The fields are typically annotated with **@Column** annotations to specify the mapping between the field and the corresponding column in the table.

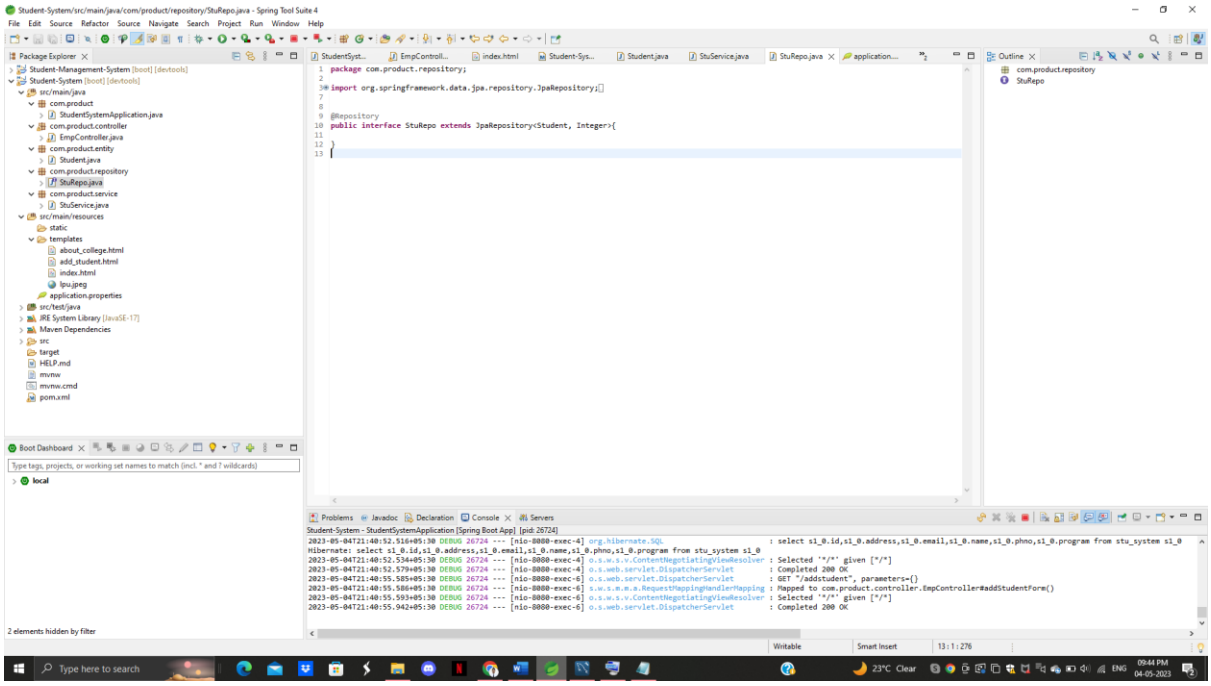




In Spring Boot, a Repository is an interface that provides a set of methods to perform CRUD (Create, Read, Update, Delete) operations on a specific type of Entity.

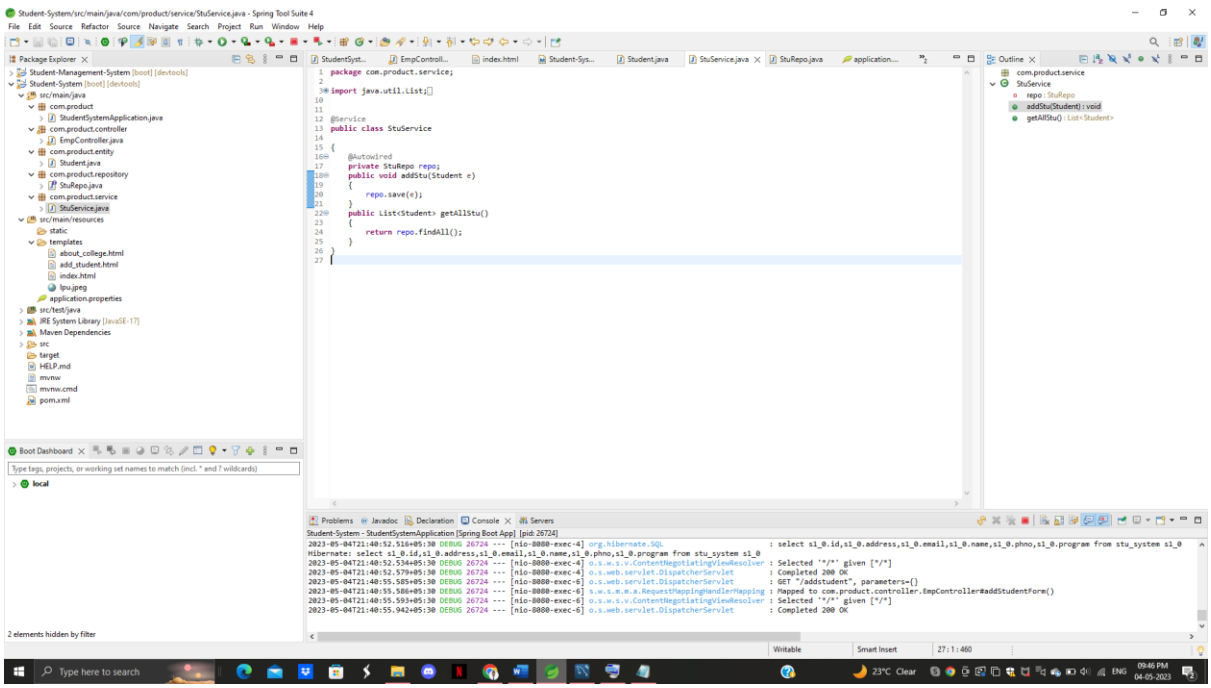
In addition to the generic methods, a Repository interface may also define custom methods to perform more specific queries or operations on the Entity type. These custom methods can be defined using Spring Data JPA's Query

Creation mechanisms, or by writing custom JPQL (Java Persistence Query Language) or native SQL queries.



In Spring Boot, a Student Service is a class that contains the business logic for managing students. It typically provides a layer of abstraction between the Controller and the Repository, and is responsible for performing validation, business logic, and calling the appropriate methods on the Repository to persist or retrieve data.

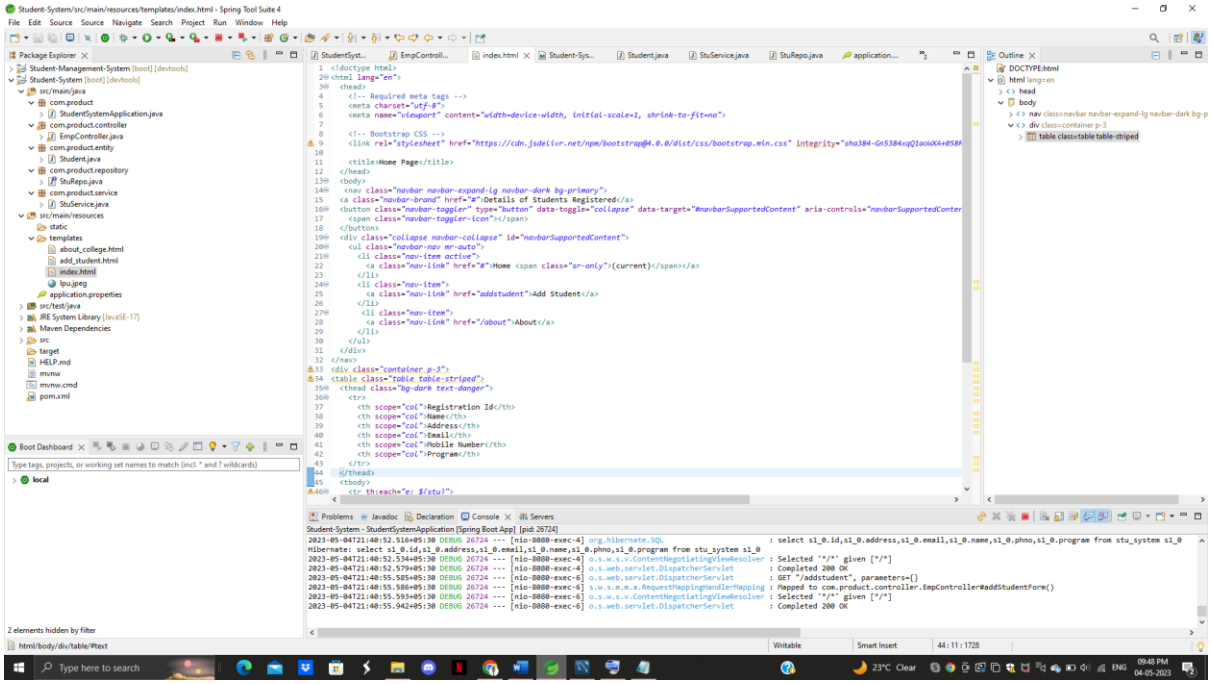
The Student Service is typically annotated with @Service to indicate that it is a Spring-managed service, and can be injected into other Spring-managed components like Controllers or other Services.

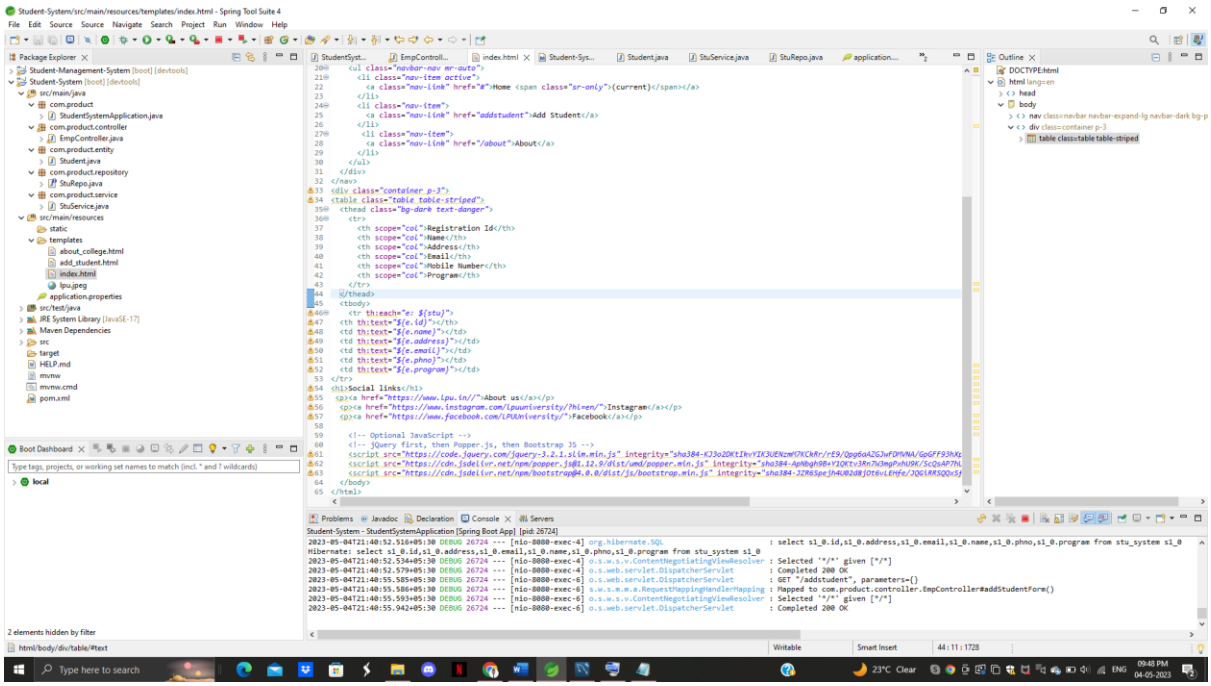


In Spring Boot, a template is a file or set of files used for generating dynamic content for web applications. These templates are typically stored in the application's resources folder and are used by Spring Boot's template engine to generate HTML, XML, or other types of output that can be sent to the browser.

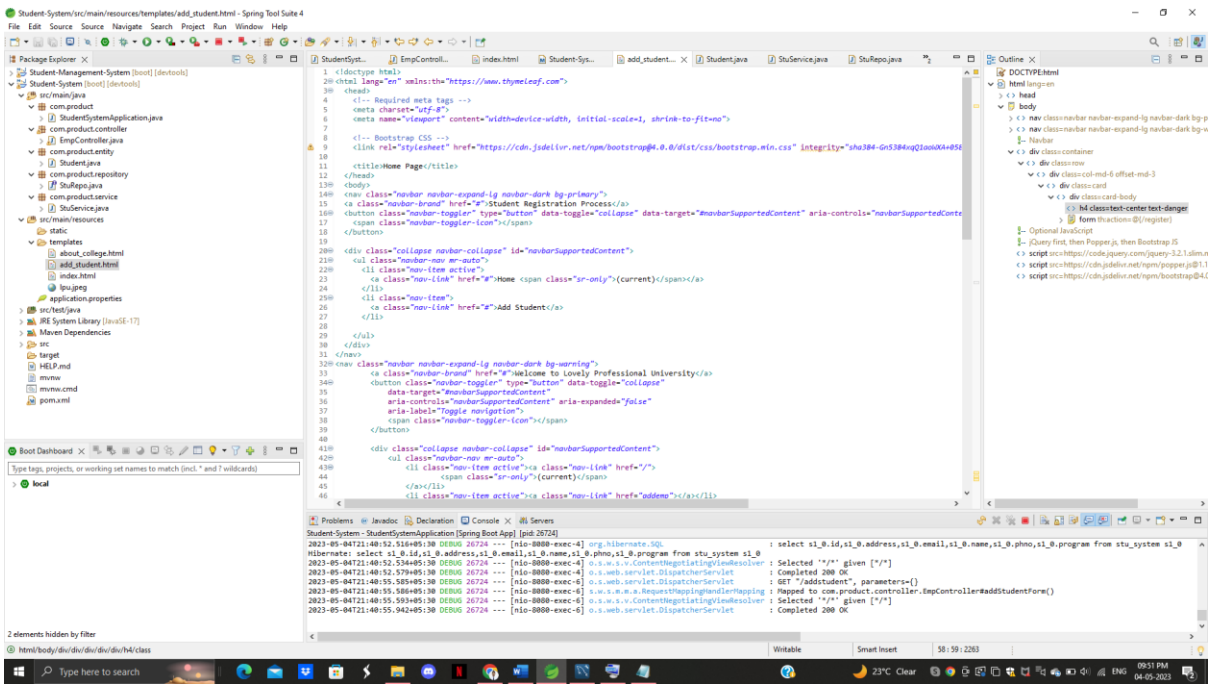
The most commonly used template engine in Spring Boot is Thymeleaf, although other engines like Freemarker and Mustache can also be used. These template engines allow developers to write templates using a markup language that is similar to HTML, but with additional syntax and features to facilitate dynamic content generation.

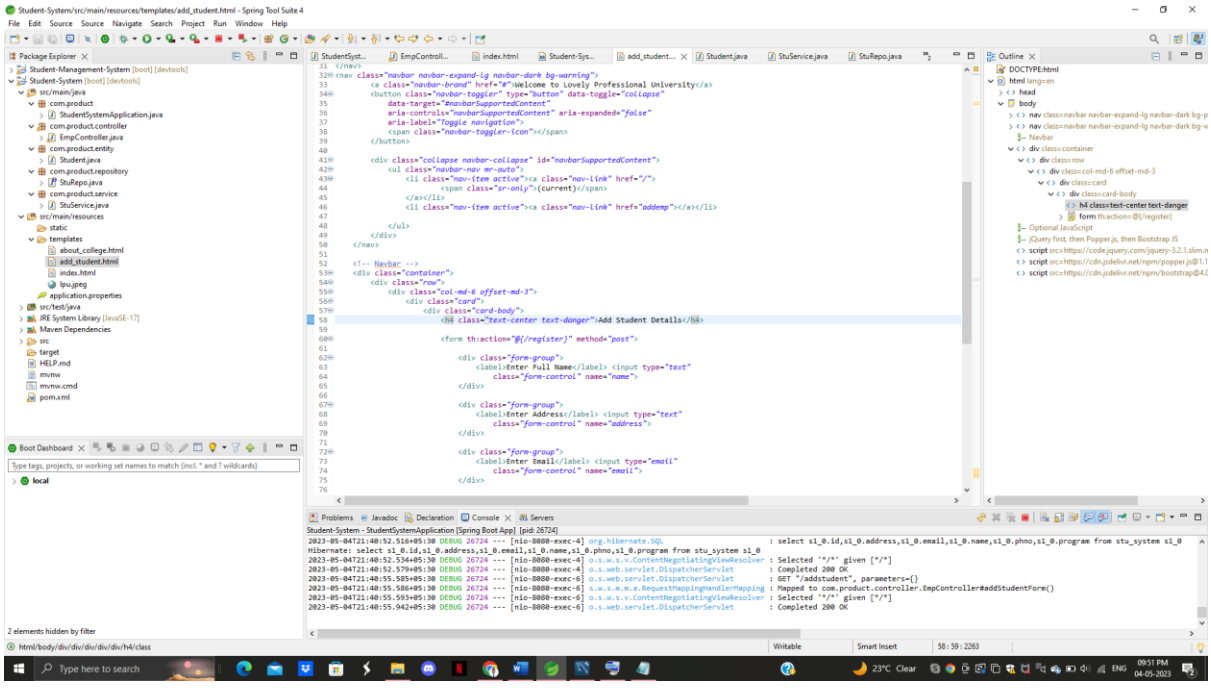
Template for the index page:-

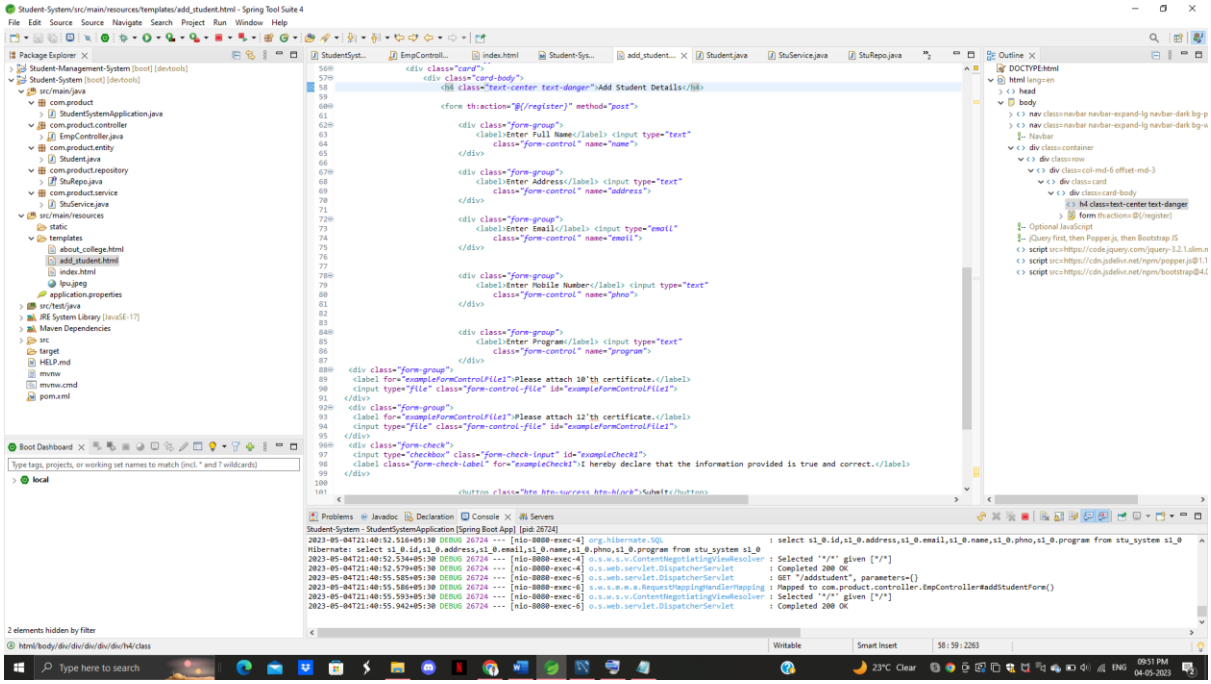


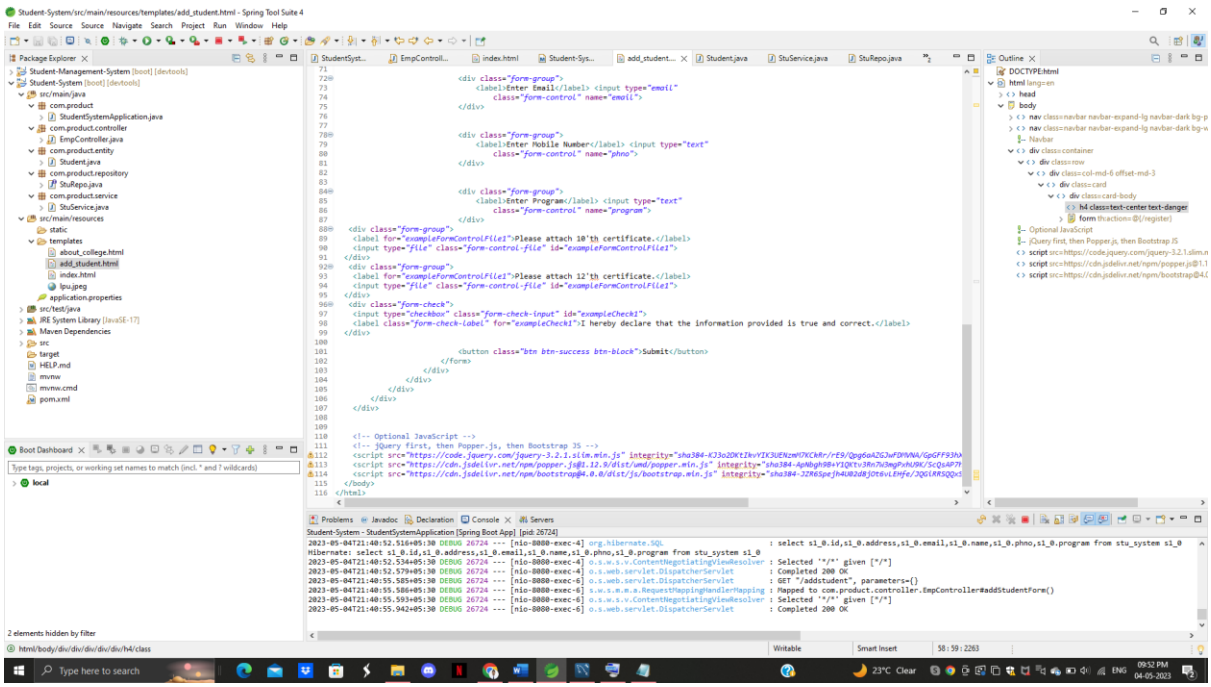


Template for adding student form :-

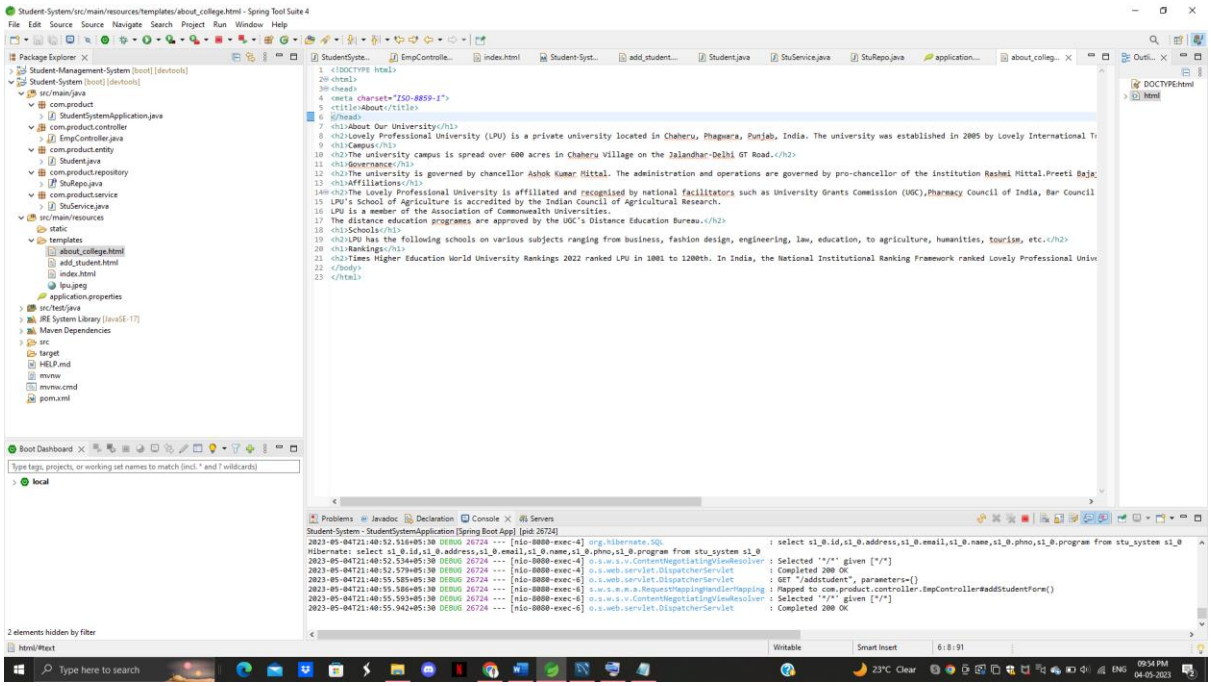





Student-System/src/main/resources/templates/add_student.html - Spring Tool Suite 4
File Edit Source Source Navigate Search Project Run Window Help
Package Explorer
Student-Management-System [boot] [devtools]
Student-System [boot] [devtools]
src/main/java
com.product
StudentSystemApplication.java
com.product.controller
EmpController.java
com.product.entity
Student.java
com.product.repository
StuRepo.java
com.product.service
StuService.java
src/main/resources
static
templates
about_college.html
add_student.html
index.html
lpu.jpeg
application.properties
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml
Boot Dashboard
local
StudentSyst...
EmpControll...
index.html
Student-Sys...
add_student...
Student.java
StuService.java
StuRepo.java
<h4 class="text-center text-danger">Add Student Details</h4>
<form th:action="@{/register}" method="post">
Enter Full Name
Enter Address
Enter Email
Enter Mobile Number
Enter Program
Please attach 10'th certificate.
Please attach 12'th certificate.
I hereby declare that the information provided is true and correct.
Submit
Outline
DOCTYPE:html
html lang=en
head
body
Navbar
Optional JavaScript
jQuery first, then Popper.js, then Bootstrap JS
Problems
Javadoc
Declaration
Console
Servers
Student-System - StudentSystemApplication [Spring Boot App] [pid: 26724]
Hibernate: select s1_0.id,s1_0.address,s1_0.email,s1_0.name,s1_0.phno,s1_0.program from stu_system s1_0
GET "/addstudent", parameters={}
Mapped to com.product.controller.EmpController#addStudentForm()
Completed 200 OK
2 elements hidden by filter
html/body/div/div/div/div/div/h4/class
Writable
Smart Insert
58 : 59 : 2263
Type here to search
23°C Clear
ENG
09:51 PM
04-05-2023


Student-System/src/main/resources/templates/add_student.html - Spring Tool Suite 4
File Edit Source Source Navigate Search Project Run Window Help
Package Explorer
Student-Management-System [boot] [devtools]
Student-System [boot] [devtools]
src/main/java
com.product
StudentSystemApplication.java
com.product.controller
EmpController.java
com.product.entity
Student.java
com.product.repository
StuRepo.java
com.product.service
StuService.java
src/main/resources
static
templates
about_college.html
add_student.html
index.html
lpu.jpeg
application.properties
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml
Boot Dashboard
local
Enter Email
Enter Mobile Number
Enter Program
Please attach 10'th certificate.
Please attach 12'th certificate.
I hereby declare that the information provided is true and correct.
Submit
Outline
DOCTYPE:html
html lang=en
head
body
Navbar
Optional JavaScript
jQuery first, then Popper.js, then Bootstrap JS
Problems
Javadoc
Declaration
Console
Servers
Student-System - StudentSystemApplication [Spring Boot App] [pid: 26724]
Hibernate: select s1_0.id,s1_0.address,s1_0.email,s1_0.name,s1_0.phno,s1_0.program from stu_system s1_0
GET "/addstudent", parameters={}
Mapped to com.product.controller.EmpController#addStudentForm()
Completed 200 OK
2 elements hidden by filter
html/body/div/div/div/div/div/h4/class
Writable
Smart Insert
58 : 59 : 2263
Type here to search
23°C Clear
ENG
09:52 PM
04-05-2023

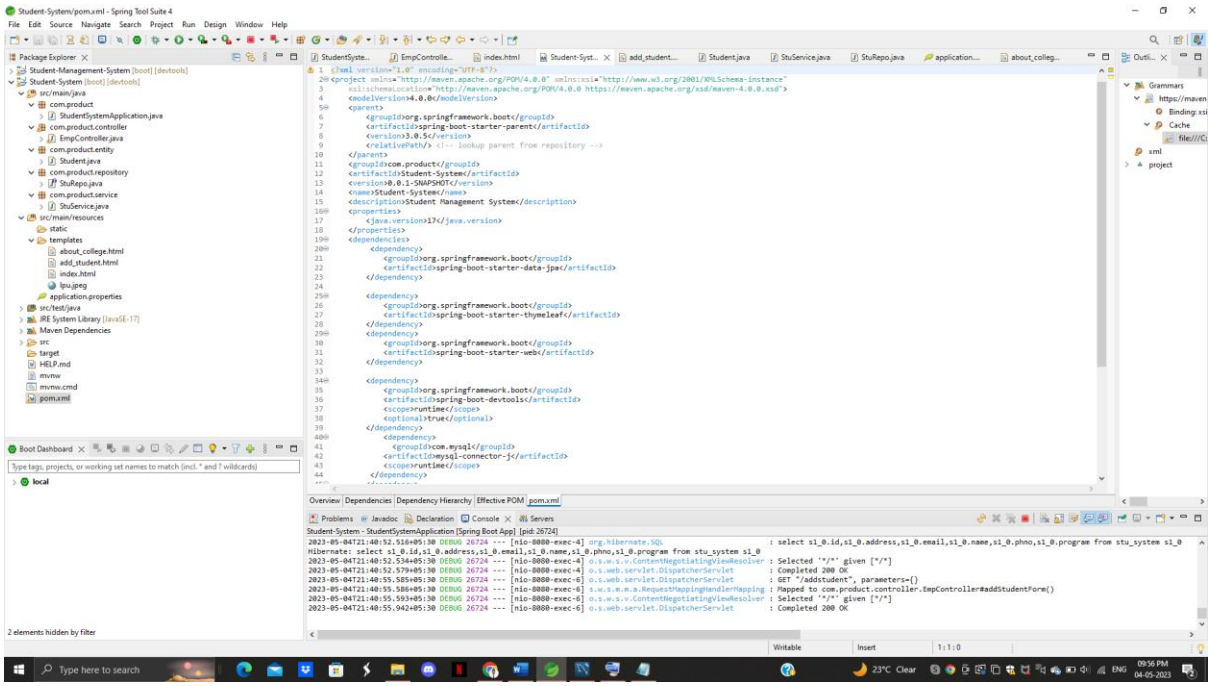
Template for about page:-



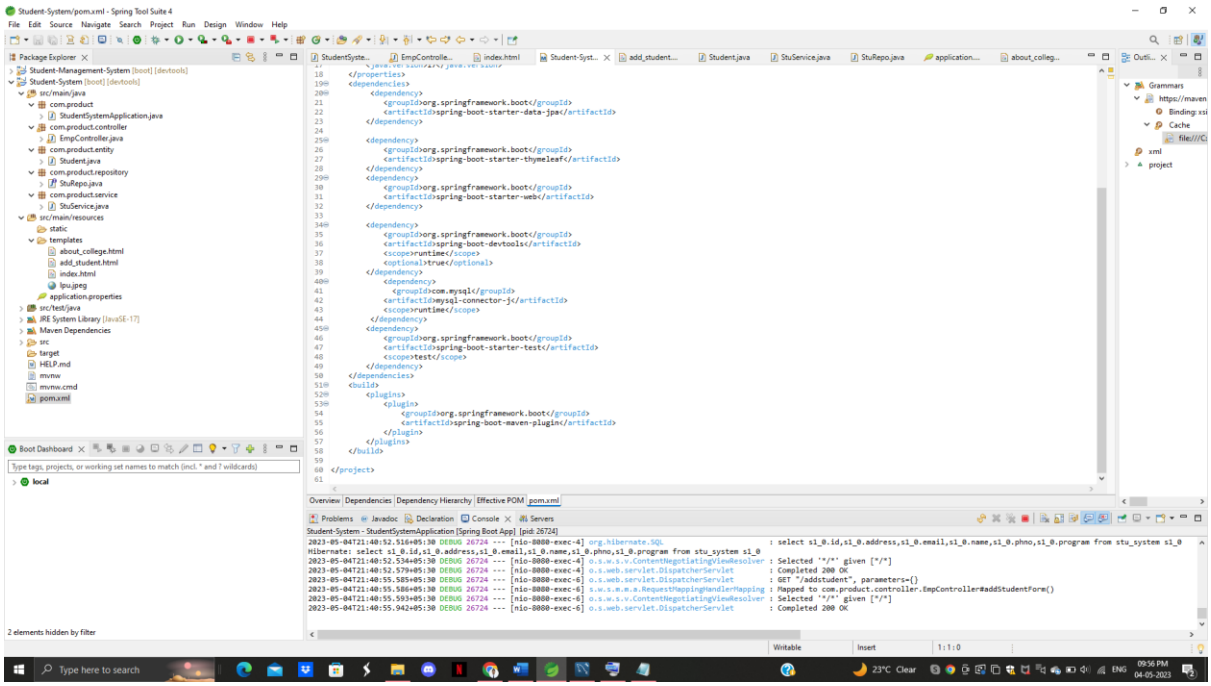
In Spring Boot, the POM (Project Object Model) file is an XML file that contains project-specific configuration and metadata for the Maven build system. The POM file is the central configuration file for a Maven-based project, and it defines the project's dependencies, build process, and other configuration options.

The POM file in a Spring Boot project specifies the parent pom and includes dependencies on the Spring Boot starter modules. The Spring Boot starter modules contain a set of pre-configured dependencies that are commonly used in Spring Boot projects, such as Spring MVC, Spring Data JPA, and Thymeleaf.

The POM file also specifies the version of Spring Boot that the project depends on, along with the versions of any other third-party dependencies that the project requires. Maven uses this information to automatically download and manage the required dependencies.



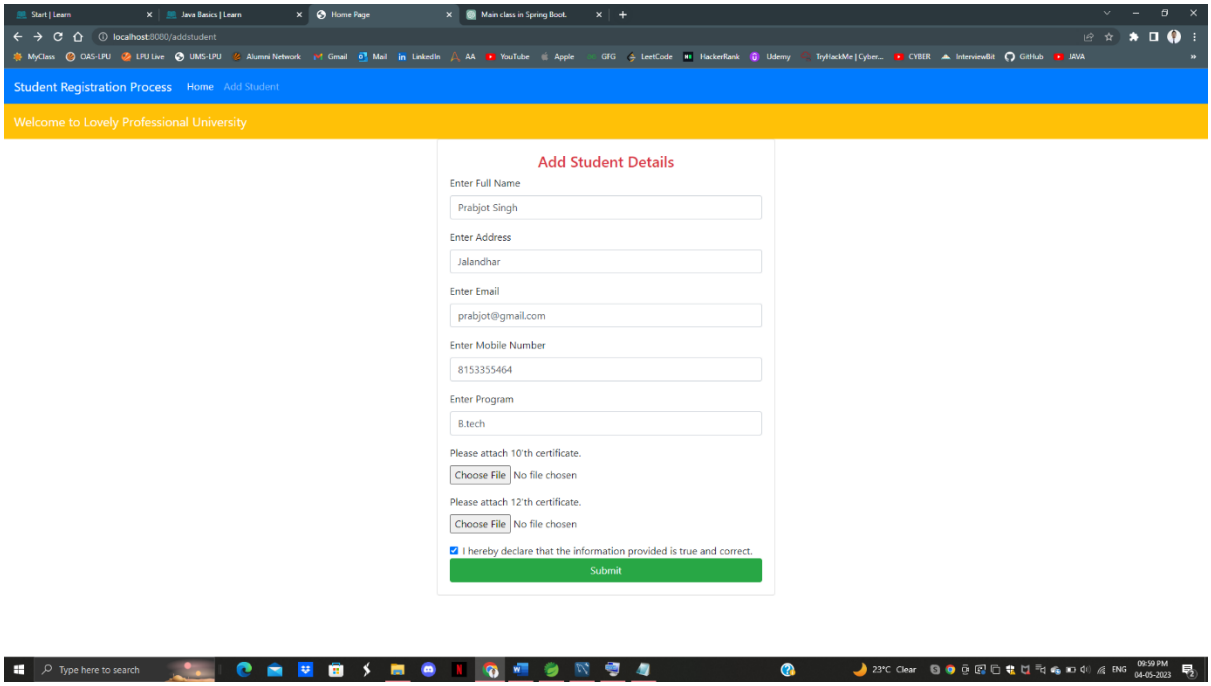
In addition to dependencies, the POM file can also contain configuration options for plugins, such as the Maven Compiler Plugin, which is responsible for compiling the project's source code. The POM file can also define project-specific properties, such as the database URL, username, and password, which can be referenced in the project's code or configuration files.



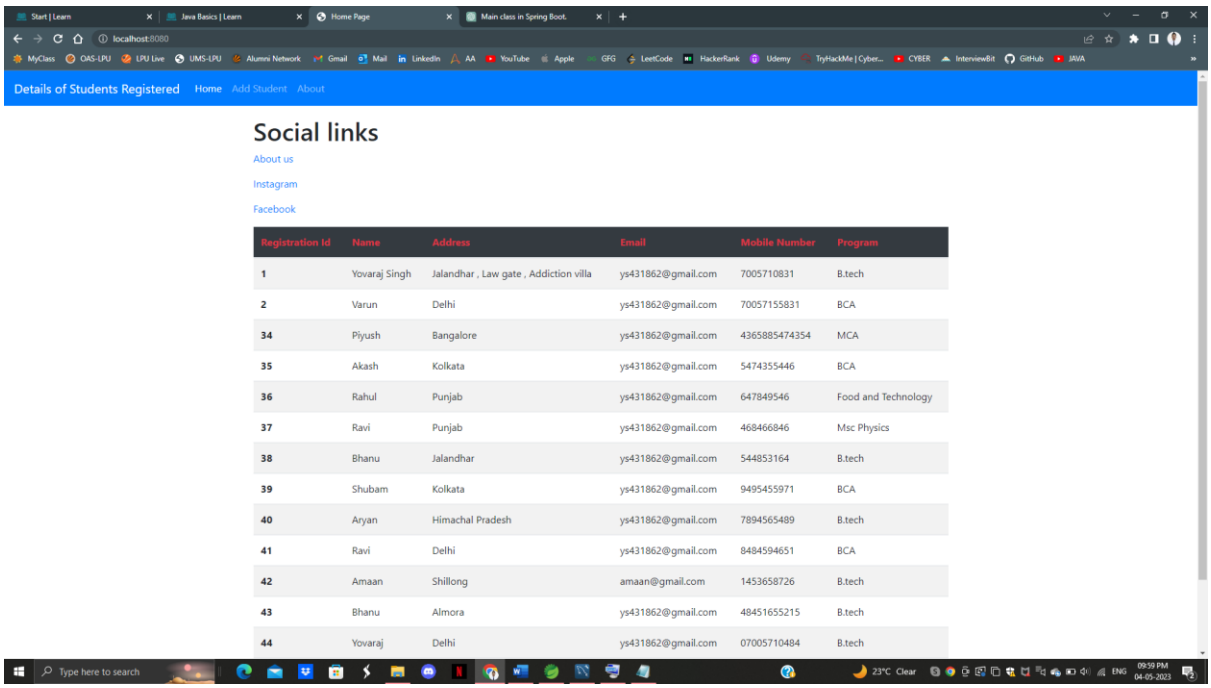
In summary, the POM file in Spring Boot is an essential configuration file that defines the project's dependencies, build process, and other configuration options. It is used by the Maven build.

Screenshot for the project:-

Entry form for students:-



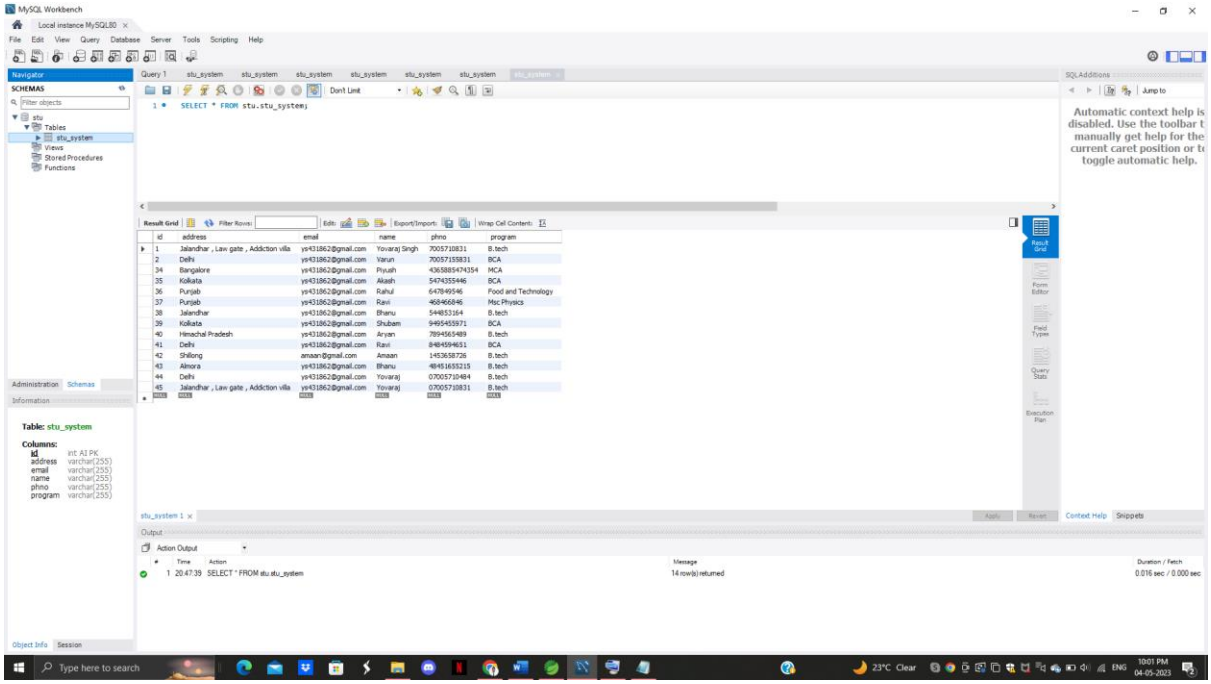
Details of the students registered:-



Registration Id	Name	Address	Email	Mobile Number	Program
1	Yovaraj Singh	Jalandhar , Law gate , Addiction villa	ys431862@gmail.com	7005710831	B.tech
2	Varun	Delhi	ys431862@gmail.com	70057155831	BCA
34	Piyush	Bangalore	ys431862@gmail.com	4365885474354	MCA
35	Akash	Kolkata	ys431862@gmail.com	5474355446	BCA
36	Rahul	Punjab	ys431862@gmail.com	647849546	Food and Technology
37	Ravi	Punjab	ys431862@gmail.com	468466846	Msc Physics
38	Bhanu	Jalandhar	ys431862@gmail.com	544853164	B.tech
39	Shubam	Kolkata	ys431862@gmail.com	9495455971	BCA
40	Aryan	Himachal Pradesh	ys431862@gmail.com	7894565489	B.tech
41	Ravi	Delhi	ys431862@gmail.com	8484594651	BCA
42	Amaan	Shillong	amaan@gmail.com	1453658726	B.tech
43	Bhanu	Almora	ys431862@gmail.com	48451655215	B.tech
44	Yovaraj	Delhi	ys431862@gmail.com	07005710484	B.tech

MySQL Workbench is a standalone application that can be used to manage MySQL databases regardless of the programming language or framework being used. However, in the context of a Spring Boot application, MySQL Workbench can be used to perform tasks such as creating and managing databases, tables, and columns, as well as executing SQL queries and managing database users and permissions.

Below is the screenshot of the MySQL Workbench for the project:-



In Spring Boot, the Application class serves as the entry point of the application. It is a Java class that is responsible for configuring and running the Spring Boot application.

The Application class in Spring Boot typically does the following:

Configures the Spring Application Context: The Application class creates and configures the Spring Application Context, which is the central container for managing the application's beans and dependencies.

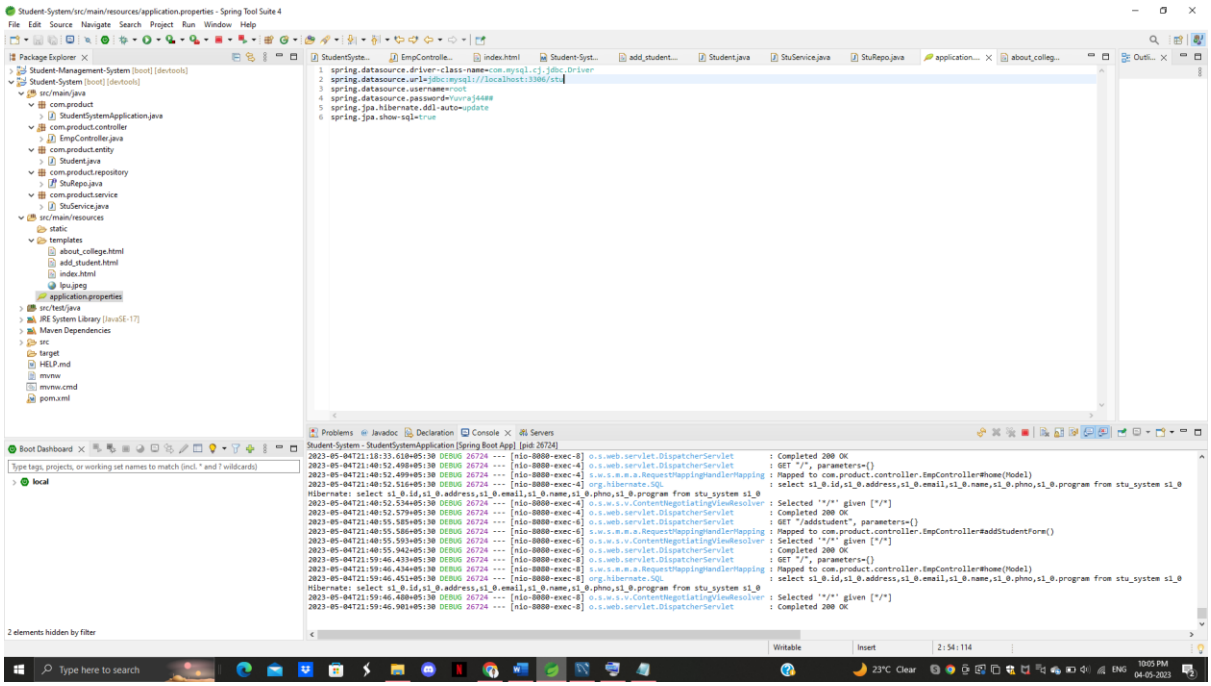
Enables Auto-Configuration: Spring Boot includes a set of pre-configured auto-configuration classes that automatically configure various aspects of the

application. The Application class is responsible for enabling these auto-configuration classes.

Starts the Embedded Web Server: Spring Boot includes an embedded web server, such as Tomcat or Jetty, which is used to serve web content. The Application class starts the embedded web server, which allows the application to receive incoming requests.

Launches the Application: Finally, the Application class launches the Spring Boot application, which makes it available to users.

Below is the screenshot of application for the project:-



Conclusion

Java is a popular programming language that is widely used in the development of web, desktop, and mobile applications. It is an object-oriented language that is designed to be platform-independent, meaning that Java code can run on any device that has a Java Virtual Machine (JVM) installed.

On the other hand, Spring Boot is a popular open-source framework that is built on top of the Java programming language. It is designed to simplify the process of building web applications and microservices, and it provides a wide range of features and tools to make development faster and more efficient.

Together, Java and Spring Boot teach us several important lessons:

Object-oriented programming: Java is designed as an object-oriented programming language, which means that it encourages developers to think in terms of objects and their interactions. This approach helps to create code that is modular, reusable, and easier to maintain.

Platform independence: Java's platform independence means that developers can write code once and run it on any device that has a JVM installed. This simplifies the development process and makes it easier to create applications that can run on multiple platforms.

Dependency injection: Spring Boot provides a powerful dependency injection framework that simplifies the process of managing dependencies between different components of an application. This approach helps to create code that is more modular, easier to test, and less tightly coupled.

Convention over configuration: Spring Boot follows the convention-over-configuration paradigm, which means that it provides sensible defaults and configurations for most common use cases. This approach simplifies the development process and reduces the amount of boilerplate code that developers need to write.

Microservices architecture: Spring Boot is often used in the development of microservices, which are small, independent services that work together to create a larger application. This approach helps to create applications that are more scalable, more fault-tolerant, and easier to maintain.

Overall, Java and Spring Boot teach us important lessons about object-oriented programming, platform independence, dependency injection, convention over configuration, and microservices architecture. These lessons can be applied to many different programming languages and frameworks, and they can help developers to create better, more efficient, and more scalable applications.

References

- https://learn.epam.com/start
- https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html
- https://spring.academy/courses
- https://spring.io/guides/gs/serving-web-content/
- https://github.com/Yovaraj