**A**

**REPORT**

**ON**

**Student Marks/Credits Management System**

**Under**

**Non-Syllabus Project (NSP)**

****

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

POORNIMA GROUP OF INSTITUTIONS

(Academic Year 2022-23(ODD))

Submitted To: - Submitted by: -

**Dr. Ajay Maurya Sahil Gupta (PGI20A033)**

**(Professor) Sourav Saini (PGI20CA036)**

**PIET(Jaipur)**

**DECLARATION**

I hereby declare that the report entitled “**STUDENT MARKS/CREDITS MANAGEMENT SYSTEM** “was carried out and written by me under the guidance of Dr. Ajay MauryaProfessor, DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE, POORNIMA GROUP OF INSTITUTIONS, Jaipur. This work has not previously formed the basis for the award of any degree or diploma or certificate nor has been submitted elsewhere for the award of any degree or diploma.

Place: PIET(JAIPUR) Sahil Gupta

Date: 24-DEC 2022 (PGI20CA033)

Sourav Saini (PGI20CA036)

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Description** | **Page Number** |
| 1. | Title | I |
| 2. | Declaration | II |
| 3. | Table of Contents | III-IV |
| 4. | List of Tables/Figures | V-VI |
| **5.** | **Chapter 1: Introduction of Project** | 1-3 |
| 6**.** | 1.1 Introduction | Page 1-2 |
| 7. | 1.2 Objective of Project | Page 2-3 |
| 8. | 1.3 Types of users | Page 3 |
| 9. | 1.4 Constraints and Dependency | Page 3 |
| 10. | 1.5 Methodology Used or Project Life cycle | Page 3 |
| **11.** | **Chapter 2: Requirement Analysis** | 4-10 |
| 12. | 2.1 Functional Requirement | Page 4-5 |
| 13. | 2.2 Non-functional Requirement | Page 5-6 |
| 14. | 2.3 Technology Used | Page 6 |
| 15. | 2.4 H/w Configuration | Page 7-8 |
| 16. | 2.5 Graphical User Interface | Page 8 |
| 17. | 2.6 Advantages | Page 8 |
| 18. | 2.7 Disadvantages | Page 8-9 |
| 19. | 2.8 Applications | Page 9 |
| 20. | 2.9 Product Perspective | Page 9 |
| 21. | 2.10 System Product Functions | Page 9-10 |
| 22. | 2.11 Advanced Features of the System. | Page 10 |
| 23. | 2.12 Operating Environment | Page 10 |
| **24.** | **Chapter3: Design** | Page 11-17 |
| 25.  26.  27.  28.  30.  31.  32. | * 1. DFD (level:0, level:1)   2. Unified Modelling Language (UML)   3. Use case Diagram   3.4 Class Diagram    3.5 Object Diagram   * 1. Sequence Diagram   3.7 Activity Diagram | Page 11  Page 12  Page 12  Page 13  Page 14  Page 15  Page 15 |

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| 32. | 3.8 Statechart Diagram | Page 16 |
| 33. | 3.9 Deployment Diagram | Page 17 |
| 34. | 3.10 Component Diagram | Page 17 |
| **35.** | **Chapter 4:**  **CONCLUSION** | Page 18 |
| **36.** | **Chapter 5:**  **REFERENCES** | Page 19 |
| **37.** | **Chapter 6:**  **SNAPSHOTS OF PROJECT** | Page 20-25 |
| **38.** | **Chapter 6:**  **SNAPSHOTS OF PROJECT** | Page 26-30 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No.** |
| 2.1 | Hardware requirements | 7 |
|  |  |  |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| 2.1 | JSP Working | 6 |
| 6 | Project screenshots | 20-30 |

**Chapter 1**

**Introduction to Student Marks Management System**

* 1. **Introduction**

This project on “Student Marks Management System” is useful for an easy user interface. The system utilizes powerful database management, data retrieval, and data manipulation. This project provides more ease for managing the data than manually maintaining the documents. The project is useful for saving valuable time and reducing the huge paperwork.

It will help educational Institutions like schools and colleges will keep track of their student records like personal details, contact details, marks details, etc. The Internet is rapidly becoming a part of the everyday lives of a majority of people in the world. People perform various activities on the Internet and one of them is storing their data in databases where they are interested. In these databases, they can post the queries and they can retrieve the required data.

There is a need for Student Information System software for the management of students’ data. There are many departments of administration for the maintenance of college information and student databases in any institution.

All these departments provide various records regarding students. Most of these track records need to maintain information about the students. This information could be the general details like student name, batch, performance, roll no., etc, or specific information related to departments like a collection of data. All the modules in college administration are interdependent. They are maintained manually.

So, they need to be automated and centralized as Information from one module will be needed by other modules. For example, when a student needs his course completion certificate it needs to check many details about the student like his name, ID, batch, marks, and many other details. So, it needs to contact all the modules that our office, department, and examination and the result of students.

**1.1.1 Basics of JSP**

JSP stands for Java Server Pages. The JSP technology can be used to create applications that generate dynamic web pages. This technology has become popular since it incorporates an important characteristic of the Java family - "Write Once, Run Anywhere".

Some of the key selling points of the JSP technology are as follows: the compilation process used produces fast results and output, and allows for more efficient websites, as the embedded code only gets compiled once and then needs mere execution before including the results in the outbound HTML file. JSP puts the static (HTML) and dynamic (mix of HTML, Java, and JSP) content into separate files. Furthermore, JSP applications feature cross-platform compatibility and can run on major web servers and operating systems.

JASPER is to be designed to show students how a real, usable program is constructed. As the intended audience for JASPER will consist of second-year computer science students (primarily CSC207 participants), special emphasis will be placed on a modular structure, straightforward implementation, and thorough documentation. The intended audience for JASPER is students with one year of university-level programming experience. Concerning programming, it is expected that the average user has a solid understanding of object-oriented programming. The users should also be able to program in an object-oriented language, preferably Java.

Documentation - extra effort will be put into the documentation of JASPER since documentation is often overdone and not useful or completely absent in beginning programmers’ code. 3 Modular design - JASPER will be designed in terms of distinct sections and modules, so that the second-year students can easily follow the program logic and obtain an understanding of the project in terms of its key components and how they come together. The intention is to produce a layer-based application wherein each layer is only marginally dependent on the others, and thus can usually function on its own. Furthermore, the modules in JASPER will tend to be somewhat smaller than those usually found in large software projects.

The level of Java will vary - in modules more easily visible and available to students it will be geared towards the student’s current level of understanding. In more encapsulated modules, or those used more infrequently, the Java programming will be more sophisticated. These design decisions will enable students to take the code apart into its component modules more easily, test it, understand it, reuse pieces as needed, and easily reassemble it into the original product

* 1. **Objective of Project**

The objective of the project is to maintain the details of marks and library details of different students. The main purpose of this project is to maintain an easy circulation system between the database and the admin, all these departments provide various records regarding students. Most of these track records need to maintain information about the students. This information could be the general details like student name, batch, performance, roll no., etc, or specific information related to departments like a collection of data.

* 1. **Types of users**

We have 3 levels of users:

* **User module**: In the user module, the user will check/update data threw GUI.
* Search student data
* Add student data
* Update student data
* Delete student data
* **Administration module:** The following are the submodule in the administration module:
* Register User
* Add Login details
  1. **Constraints and Dependency**

As in any real software project, several different risks became tangible during the development of JASPER. The key factors impacting the discrepancy between the original design and the final delivered application are the following:

 Schedule setbacks due to small bugs and module integration

 The expected functionality of external packages differed from the actual functionality available

 Misleading or lacking documentation hindered the reuse of required packages

 The time required for individual system components was sometimes underestimated

 Project progress was at times ambiguous, due to the famous “90% complete” syndrome

* 1. **Methodology Used or Project Life cycle**

The methodology used in the formation of this system is Waterfall Model, Software Development Life Cycle Model (SDLC), in which tasks are divided into units and each unit is integrated into every next step in the software development process.

**Chapter 2**

**Requirement Analysis**

* 1. **Functional Requirement**
* 2.1.1: **Login**.
* Regular person: The user has to provide details about his/her User ID and password.
* Input: Enter the User ID and password provided.
* Output: The user will be able to use the features of the software.
* 2.1.2: **Add student details by User**
* 2.1.2.1**: Add whole new details of the student**
* Description: The user has to enter details like student id, name, batch, age, roll no., marks, etc.
* 2.1.2.2: **Add new course data to current student**
* State: Add course data to existing students.
* Input: course id, marks.
* Output: confirmation or an apology for failure in the issue.
* 2.1.3: **Update student details by User**
* 2.1.3.1: **update Details of student**
* State: update data to existing students.
* Input: student id.
* Output: confirmation or an apology for failure in the issue.
* 2.1.4: **Delete student details by User**
* 2.1.4.1:  **delete Details of student**
* State: delete data to existing students.
* Input: student id.
* Output: confirmation or an apology for failure in the issue.
* 2.1.5: **view student details by User**
* 2.1.5.1:  **view Details of the student**
* State: existing data to existing students.
* Input: student id.
* Output: confirmation or an apology for failure in the issue.
* 2.1.5.2:  **view Details of all the student**
* State: existing data to existing students.
* Output: Detailed table.

**2.2 Non-functional Requirement**

* **Usability Requirement**

The system shall allow the users to access the system from the web browser. The system uses a website as an interface. Since all users are familiar with the general usage of the websites, no special training is required. The system is user-friendly which makes the system easy.

* **Availability Requirement**

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

* **Efficiency Requirement**

Mean Time to Repair (MTTR) - Even if the system fails, the system will be recovered back up within an hour or less.

* **Accuracy**

The system should accurately provide real-time information taking into consideration various concurrency issues. The system shall provide 100% access reliability.

* **Performance Requirement**

The information is refreshed depending on whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

* **Reliability Requirement**

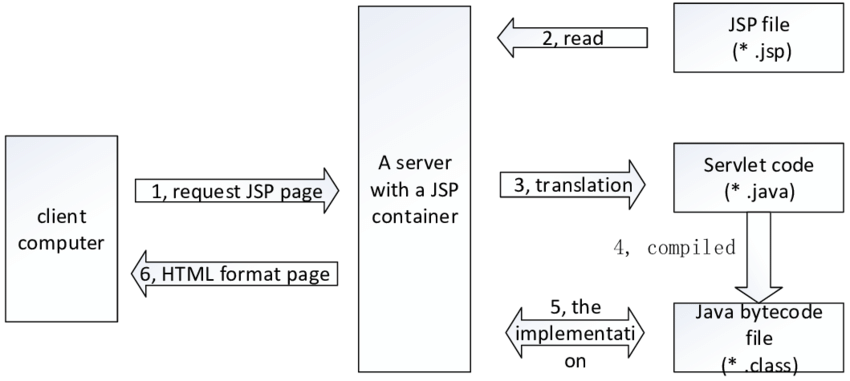
The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week, 24 hours a day.

* 1. **Technology Used**

**2.3.1 Basics of JSP**

JSP stands for Java Server Pages. The JSP technology can be used to create applications that generate dynamic web pages. This technology has become popular since it incorporates an important characteristic of the Java family - "Write Once, Run Anywhere".

Some of the key selling points of the JSP technology are as follows: the compilation process used produces fast results and output, and allows for more efficient websites, as the embedded code only gets compiled once and then needs mere execution before including the results in the outbound HTML file. JSP puts the static (HTML) and dynamic (mix of HTML, Java, and JSP) content into separate files. Furthermore, JSP applications feature cross-platform compatibility and can run on major web servers and operating systems.

JASPER is to be designed to show students how a real, usable program is constructed. As the intended audience for JASPER will consist of second-year computer science students, special emphasis will be placed on a modular structure, straightforward implementation, and thorough documentation. The intended audience for JASPER is students with one year of university-level programming experience. Concerning programming, it is expected that the average user has a solid understanding of object-oriented programming. The users should also be able to program in an object-oriented language, preferably Java

|  |
| --- |
| Figure 2.1 |

**2.4 H/w Configuration**

|  |
| --- |
| Table 2.1 |

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Hardware used** | **Specification** |
| **1.** | Monitor | LCD 15.6” screen (DELL) |
| **2.** | Keyboard | DELL Wireless |
| **3.** | Mouse | DELL Wireless |
| **4.** | Hard drive | 1TB(Terabyte) hard disc drive and 256GB(gigabyte) Solid-state drive |
| **5.** | Ram | 8GB(gigabyte) |
| **6.** | System type | 64-bit operating system, x64-based processor |
| **7.** | Processor | 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 3.00 GHz |
| **8.** | Graphics: | Onboard graphics card,2GB(Megabyte of memory) |
| **9.** | Device ID | A104FCA4-2C2E-4234-BAA0-9D28878F5BD4 |

**2.5 Graphical User Interface**

The original design permits the user to exercise multiple command-line options leading to somewhat different results in JSP page compilation and display. Although user input is correctly parsed for such options, not all such options are currently available to the user. For instance, since our primary focus was to implement JASPER as a complete JSP server engine, specifying the file name to be compiled as a command-line argument will currently not affect JSP page parsing and

execution. Only when the page is requested through the browser proper will JASPER detect the requested file and process it accordingly. Another aspect of the user interface that was implemented to a limited extent is error handling. JASPER will detect errors such as improper execution flags or incorrect file types to be compiled (such as a .asp file). As well, if a file cannot be found, the standard Error 404 – File not found is displayed to the browser. However, no user-friendly errors were implemented for faulty JSP code embedded within a page, or for pages that are not compliant with proper HTML and XHTML standards. Rather than allowing for a graceful recovery, the program may display an error message that will be confusing or incomprehensible to the user.

**2.6 Advantages**

The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or another MS-specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.

* 1. **Disadvantages**
* It is very difficult for developers to perform database connectivity in JSP.
* As the JSP is compiled on the server, it is not memory and time-efficient.
* It is hard to track errors in JSP files because they are an extension to Servlets. The JSP codes are processed into Servlet codes for compilation.
* As JSP is an HTML file, it doesn’t provide many features.
  1. **Applications**

In the early days of the Web, the Common Gateway Interface (CGI) was the only tool for developing dynamic web content. However, CGI is not an efficient solution. For every request that comes in, the web server has to create a new operating system process, load an interpreter and a script, execute the script, and then tear it all down again. This is very taxing for the server and Doesn’t scale well when the number of traffic increases.

Numerous CGI alternatives and enhancements, such as Fast CGI, mod\_ Perl from Apache, NSAPI from Netscape, ISAPI from Microsoft, and Java Servlets from Sun Microsystems, have been created over the years. While these solutions offer better performance and scalability, all of these technologies suffer from a common problem: they generate web pages by embedding HTML directly in programming language code. This pushes the creation of dynamic web pages exclusively into the realm of programmers. Java Server Pages, however, changes all that.

**2.10 System Product Functions**

JASPER requires a class loader because a user could put a reference to a new dynamically loaded class inside their JSP code. The class loader will load classes that are used in the future by the program based on the currently running class. The class loader is responsible for importing the binary data that defines the classes and interfaces of the running program. Another useful aspect of the class loader is that it provides some security to JASPER. The class loader provides security as follows: it keeps separate, code downloaded from different sources and prevents the loading of classes that declare themselves part of a trusted library, but which are, in fact, untrusted.

**2.11 Advanced Features of the System**

The Web has evolved from a network-based hypermedia distributed information system offering static information to a marketplace for selling and buying goods and services. The increasingly sophisticated applications enable this marketplace to require technology for presenting dynamic information. First-generation solutions included CGI, which is a mechanism for running external programs through a web server. The problem with CGI scripts is scalability; a new process is created for every request. Second-generation solutions included web server vendors providing plug-ins and APIs for their servers.

The problem is that their solutions were specific to their server products. For example, Microsoft provided Active Server Pages (ASP) that made it easier to create dynamic content. However, their solution only worked with Microsoft IIS or Personal Web Server. Therefore, if you wanted to use ASP you had to commit yourself to Microsoft products and you would not be enjoying the freedom of selecting your favorite web server and operating system! Another second-generation technology that is quite popular in enterprise computing is servlets.

Servlets make it easier to write server-side applications using Java technology. The problem with either CGI or servlets, however, is that you have to follow the write, compile, and deploy life cycle. JSP pages are a third-generation solution that can be combined easily with some second-generation solutions, creating dynamic content, and making it easier and faster to build web-based applications that work with a variety of other technologies: web servers, web browsers, application servers, and other development tools.

**2.12 Operating Environment**

To run JSP pages, you need a web server with a web container that conforms to JSP and servlet specifications. The web container executes on the web server and manages the execution of all JSP pages and servlets running on that web server. Tomcat 3.2.1 is a complete reference implementation for the Java Servlet 2.2 and JSP 1.1 specifications. Download and install binary versions of Tomcat. To configure Tomcat:

• Set the environment variable JAVA\_HOME to point to the root directory of your Java 2 Standard Edition(J2SE)installation. • Set the TOMCAT\_HOME environment variable to point to the root directory of your Tomcat installation.

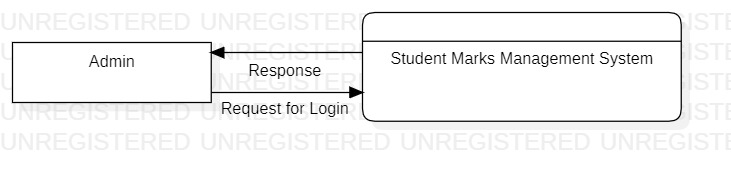
• To start Tomcat, use TOMCAT\_HOME/bin/startup.bat for windows or startup. sh for UNIX. By default, it will start listening on port 8080. • Save your .jsp files in TOMCAT\_HOME/webapps/examples/jsp and your JavaBeans classes in TOMCAT\_HOME/webapps/examples/web-inf/classes.

**Chapter 3**

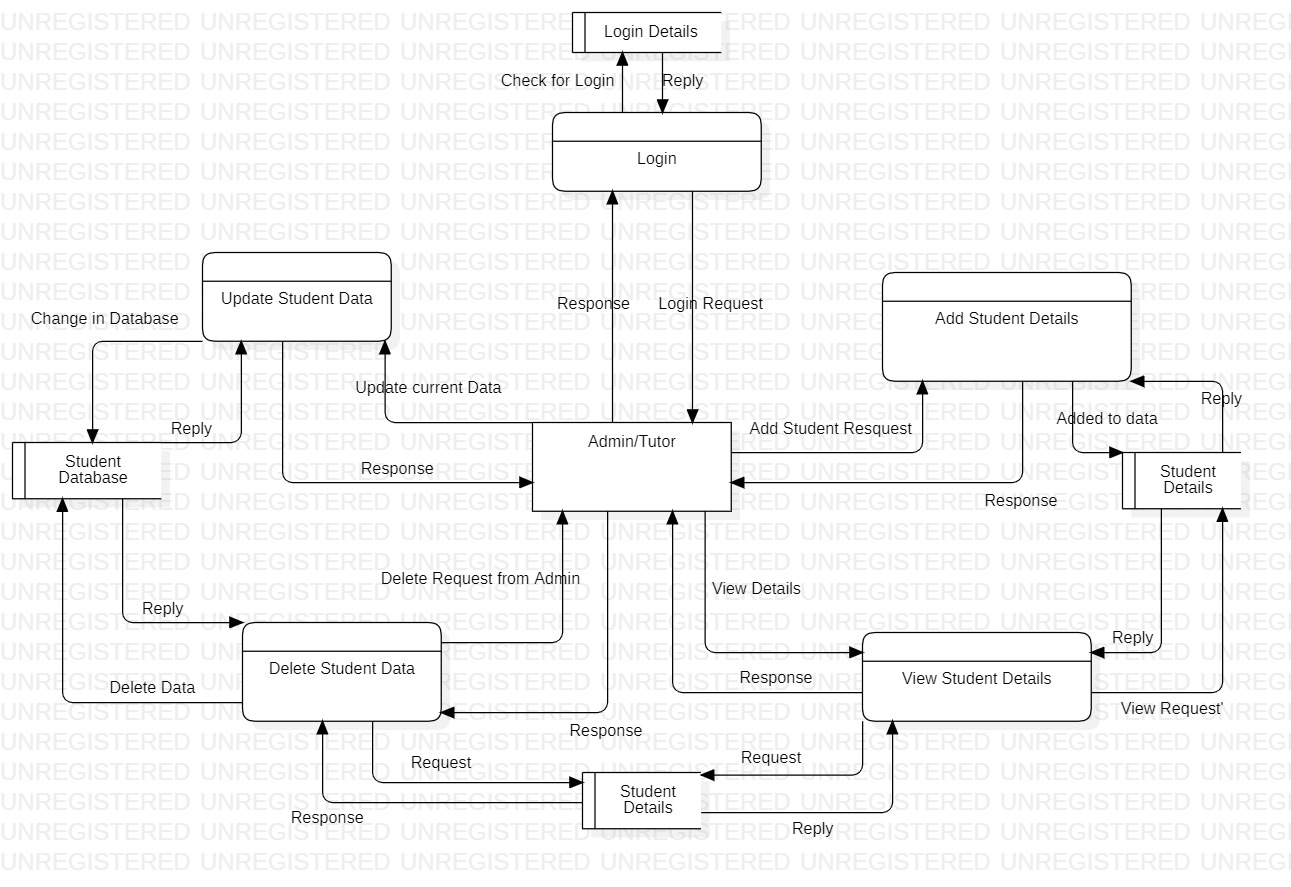
**DESIGN**

**3.1 Data Flow Diagram (DFD)**

**3.1.1 Level 0**

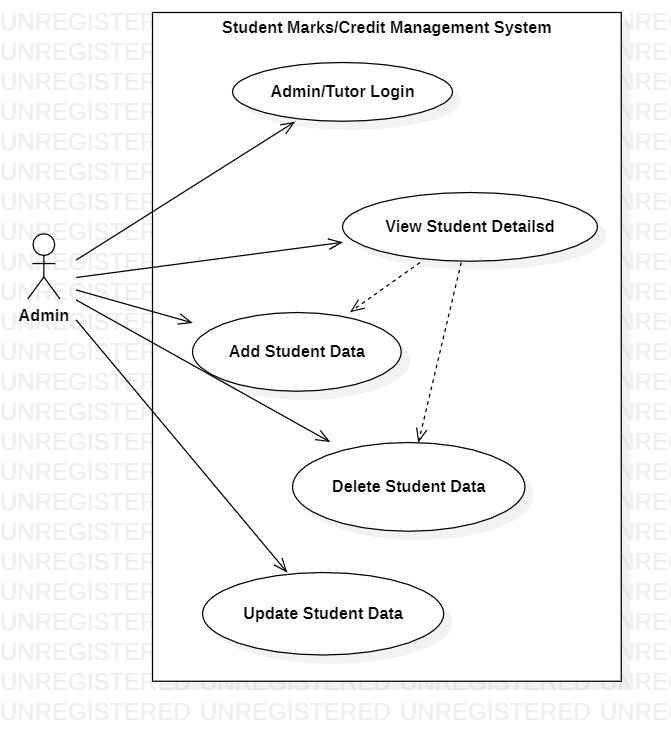


**3.1.3 Level 1**

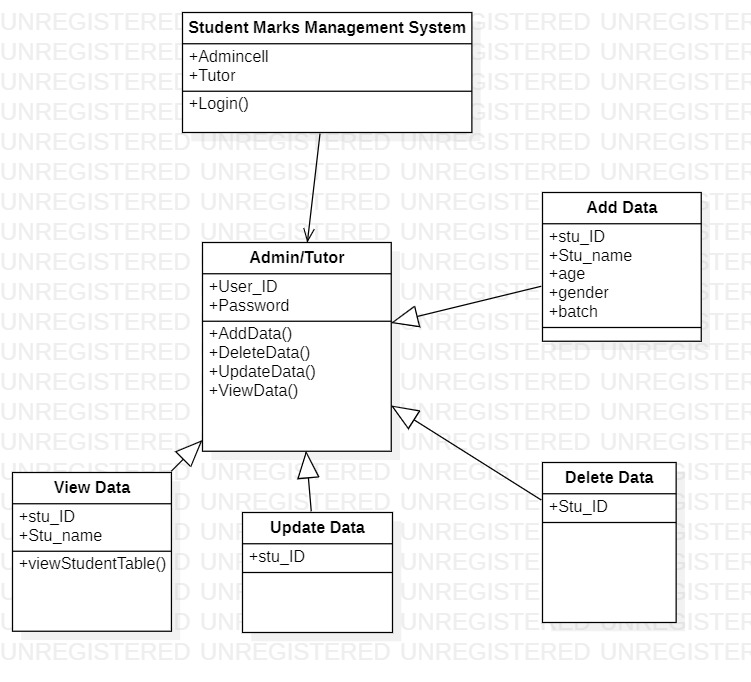


* 1. **Unified Modelling Language (UML)**

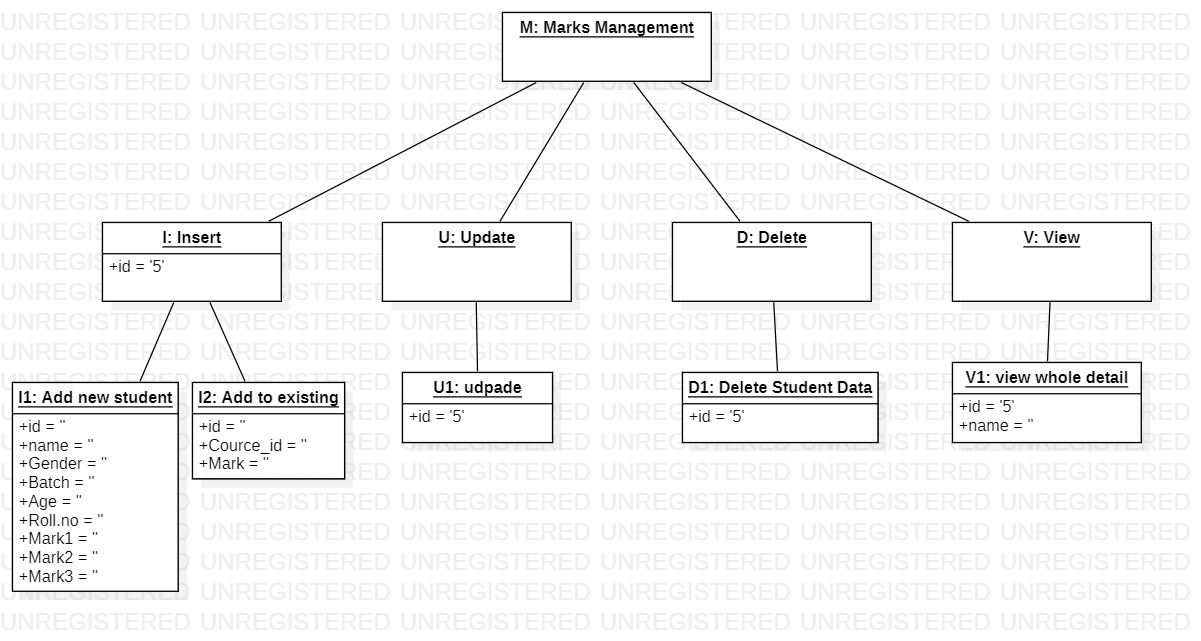
**3.3 Use case Diagram**



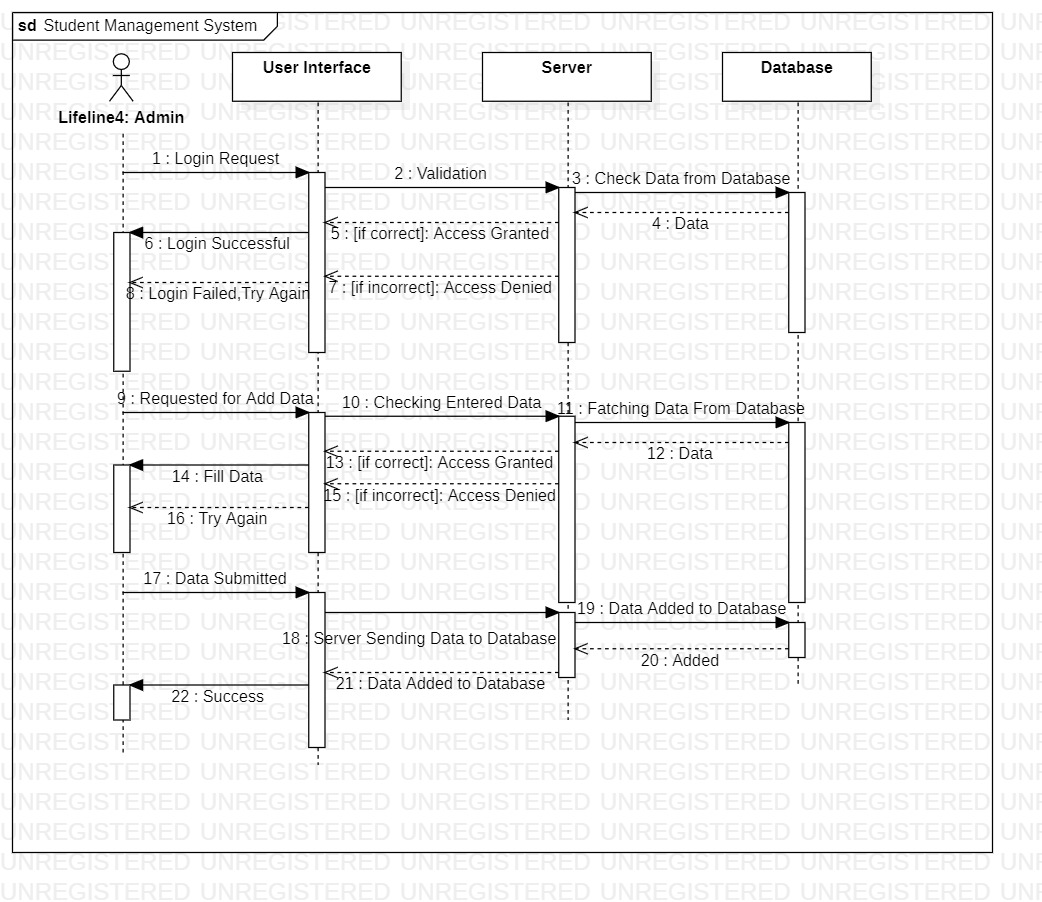
* 1. **Class Diagram**



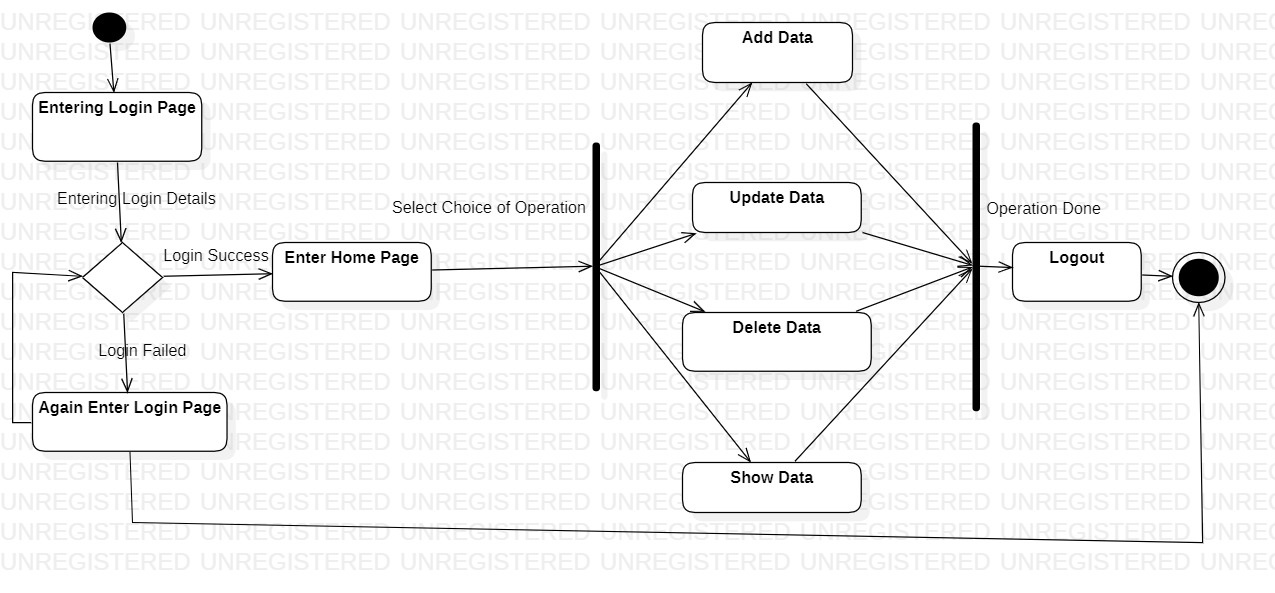
**3.5 Object Diagram**



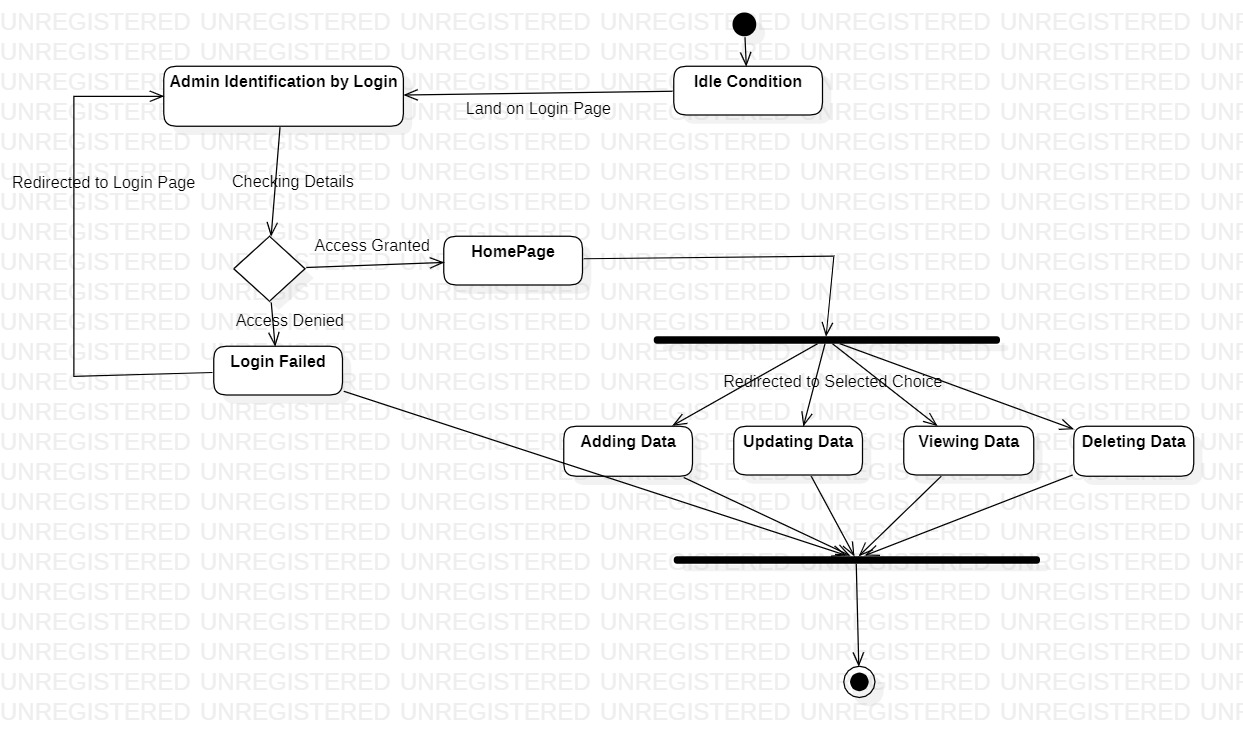
**3.6 Sequence Diagram**

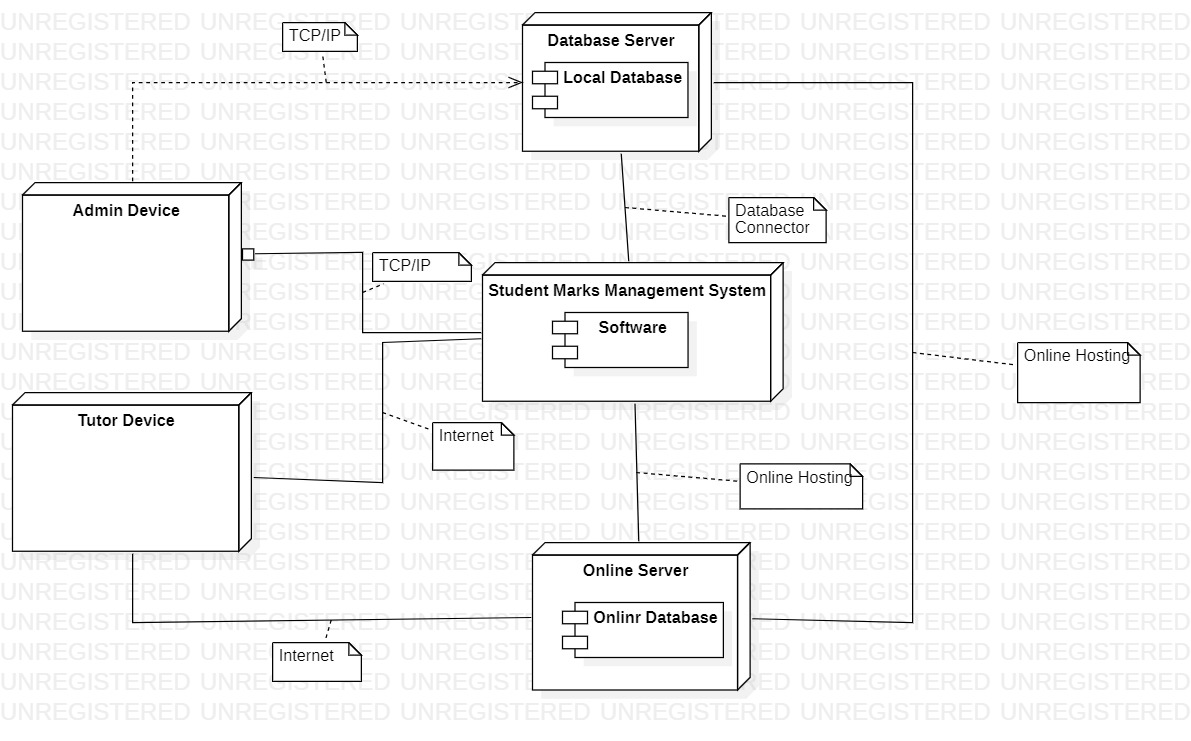


**3.7 Activity Diagram**

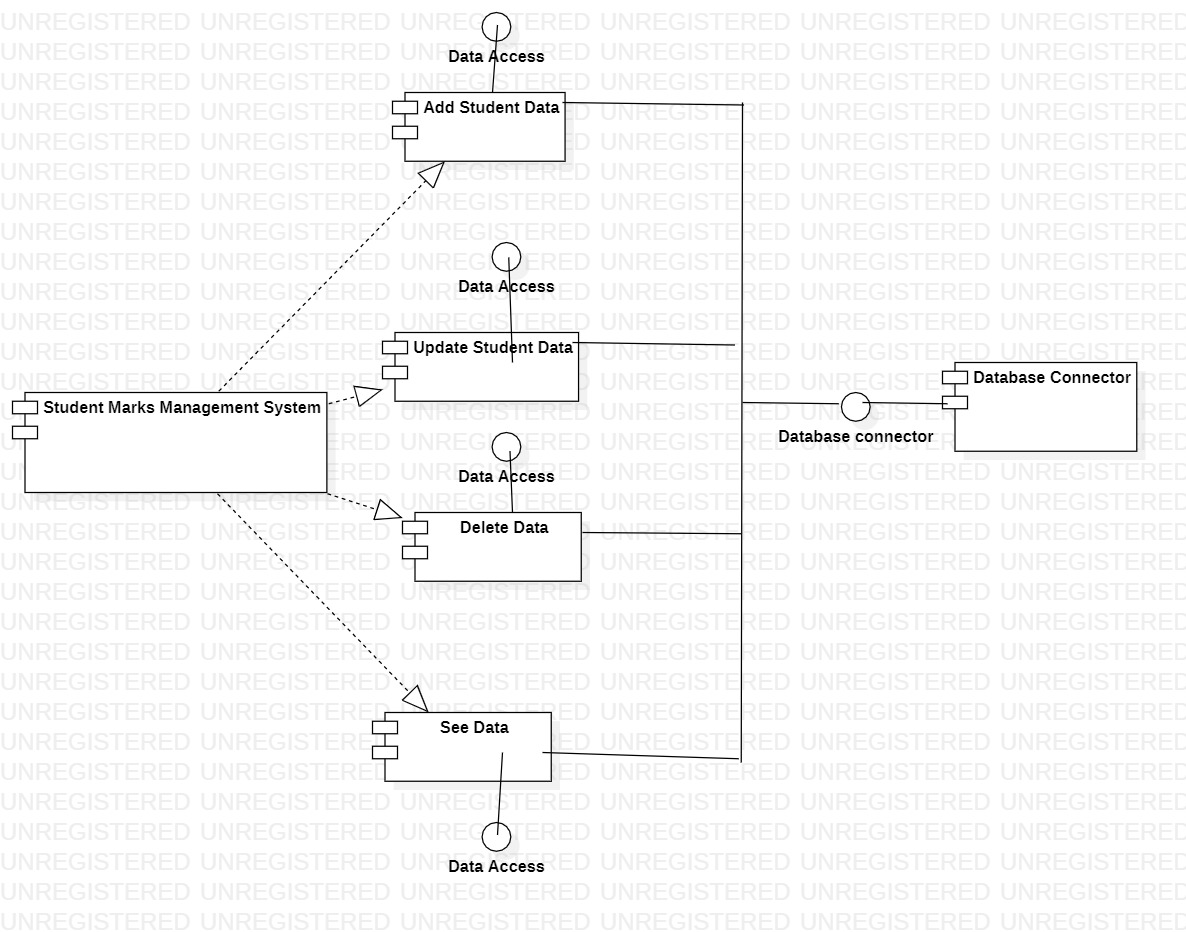


**3.8 Statechart Diagram**



* 1. **Deployment Diagram**

**3.10 Component Diagram**



**Chapter 4**

**CONCLUSION**

The most convenient aspect of the JASPER project would have been if the JAXP parser parsed HTML and JSP code. This would eliminate the need for costly workarounds and ensure that proper parsing was done concerning all the specific JSP details.

Another useful feature would be a simple API allowing JASPER to access the underlying file system, irrespective of the operating system.

The primary payoff would be the flexibility with which the HTTP listener and class loader modules could locate and access requested files, without special modification. Another important payoff factor would be the dissociation of JASPER from a specific operating system.

Such system independence could ease future development efforts, as well as eliminate a possible source of integration difficulty if JASPER was to be run on a different system.

A final important component that could be useful for JASPER is a simple GUI interface for the user, allowing actions such as starting and stopping JASPER, pre-specifying default directories that the user wished to use, and perhaps adding new JSP files to be processed by JASPER in the future.

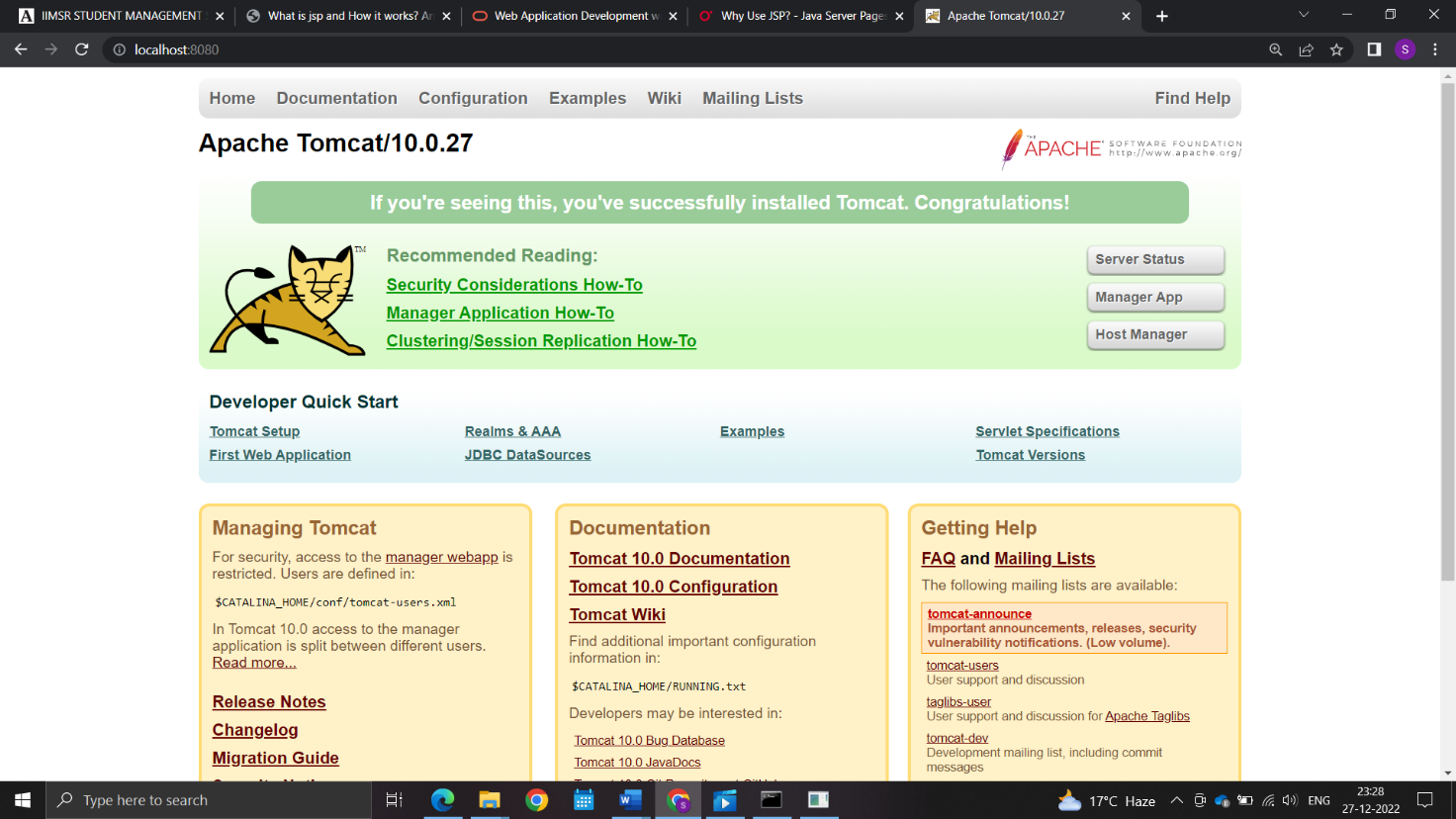
**Chapter 5**

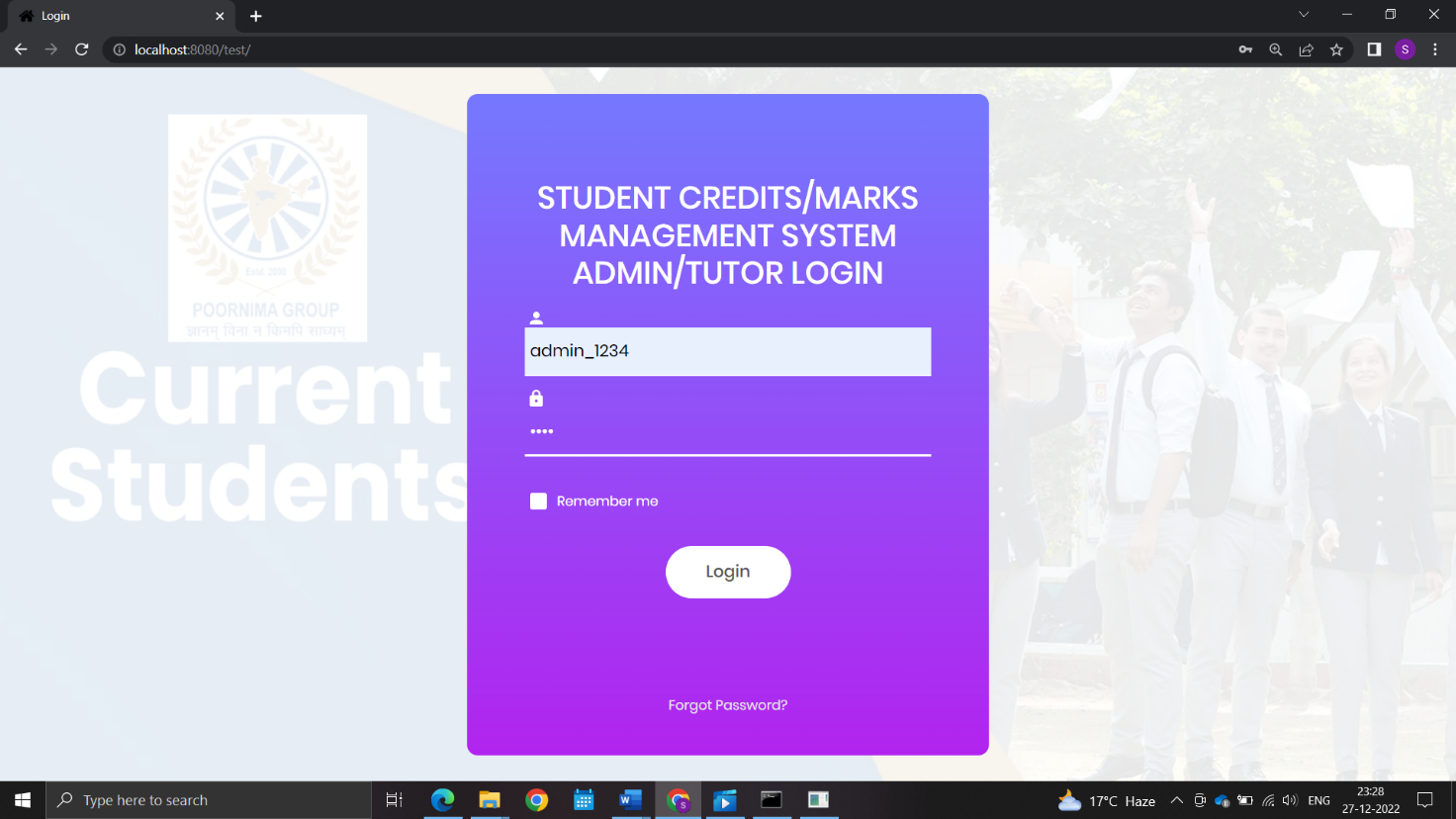
**REFERENCES**

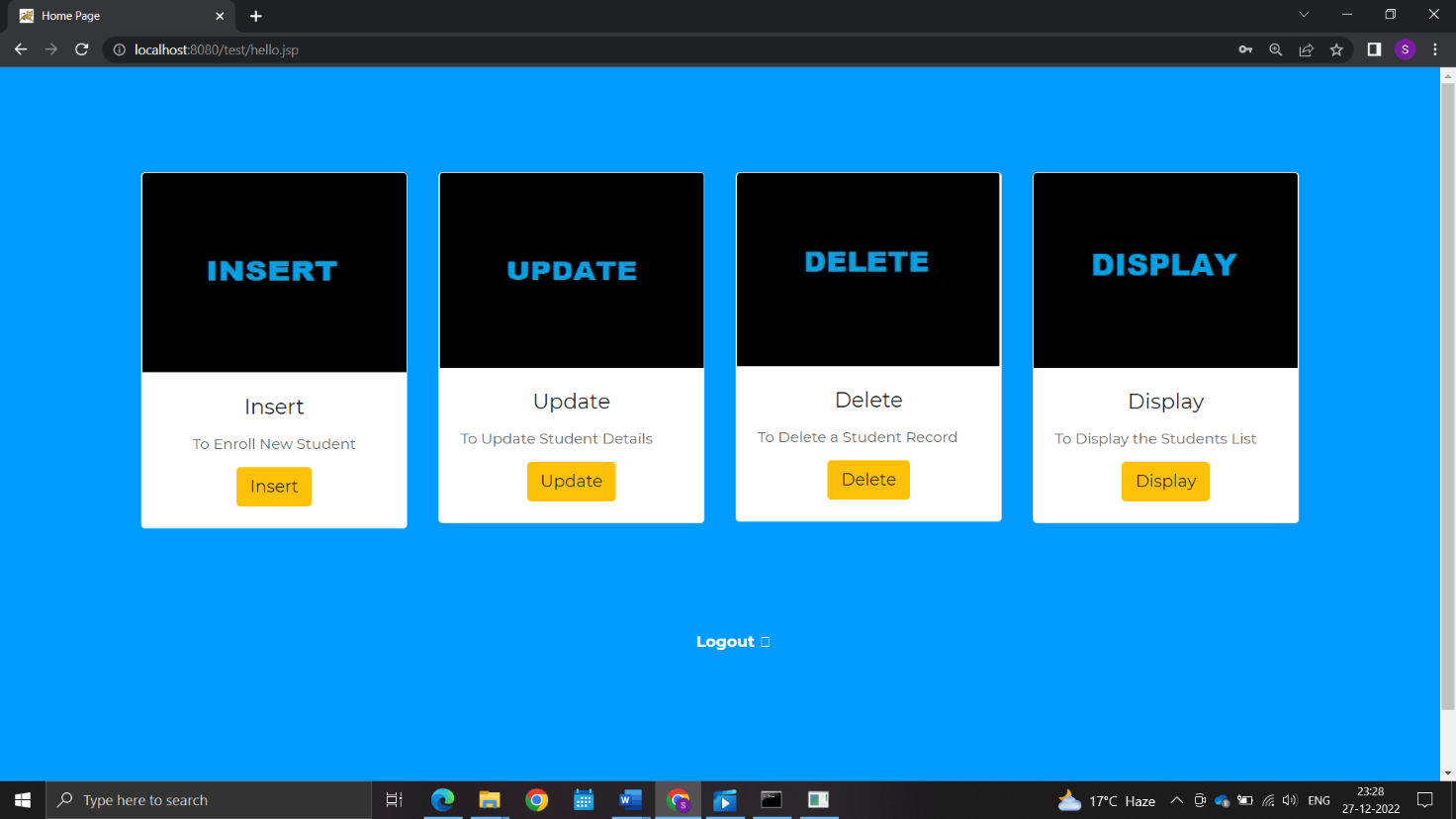
1. <https://www.oracle.com/technical-resources/articles/javase/webappdev.html>
2. <https://www.oreilly.com/library/view/java-server-pages/156592746X/ch01s02.html>
3. <https://www.academia.edu/34508514/IIMSR_STUDENT_MANAGEMENT_SYSTEM_A_PROJECT_REPORT>
4. <https://ducmanhphan.github.io/2019-01-25-How-jsp-works/>
5. <https://www.researchgate.net/profile/Marc-Denecker/publication/2905759_Project_Report_on/links/543bb9170cf204cab1db1e1b/Project-Report-on.pdf?origin=publication_detail>
6. <https://www.devopsschool.com/blog/what-is-jsp-and-how-it-works-an-overview-and-its-use-cases/>

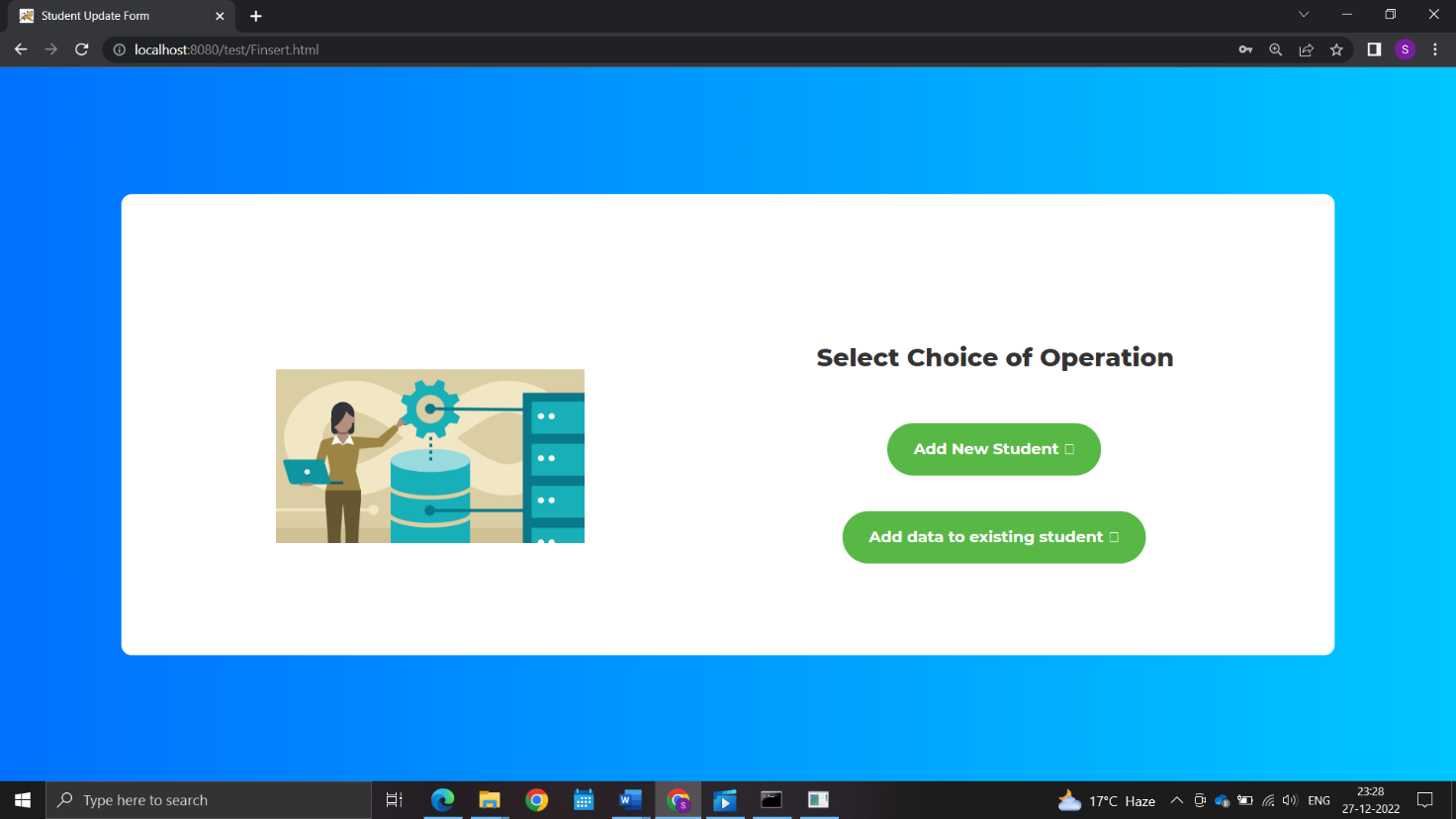
**Chapter 6**

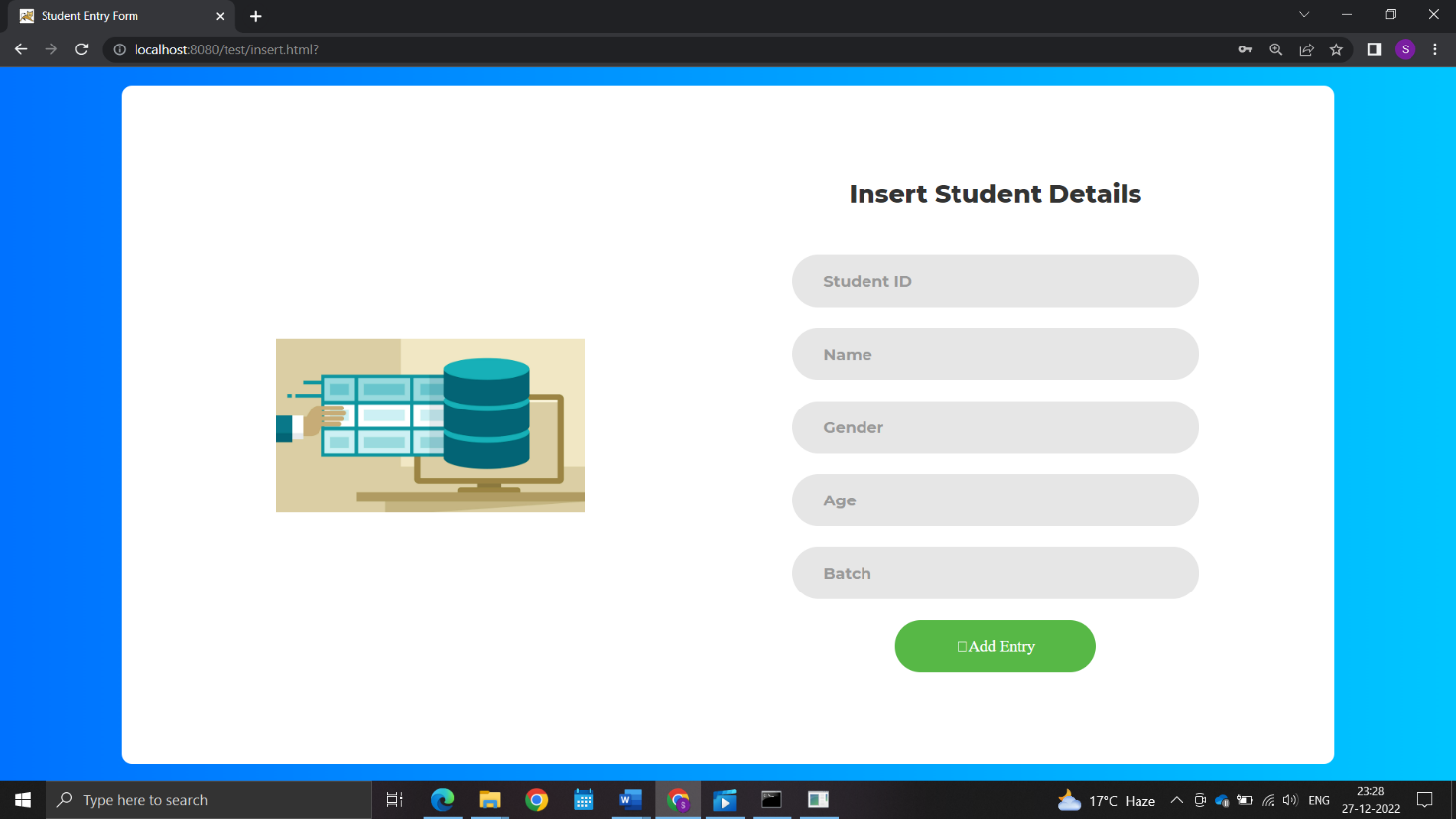
**SNAPSHOTS OF PROJECT**

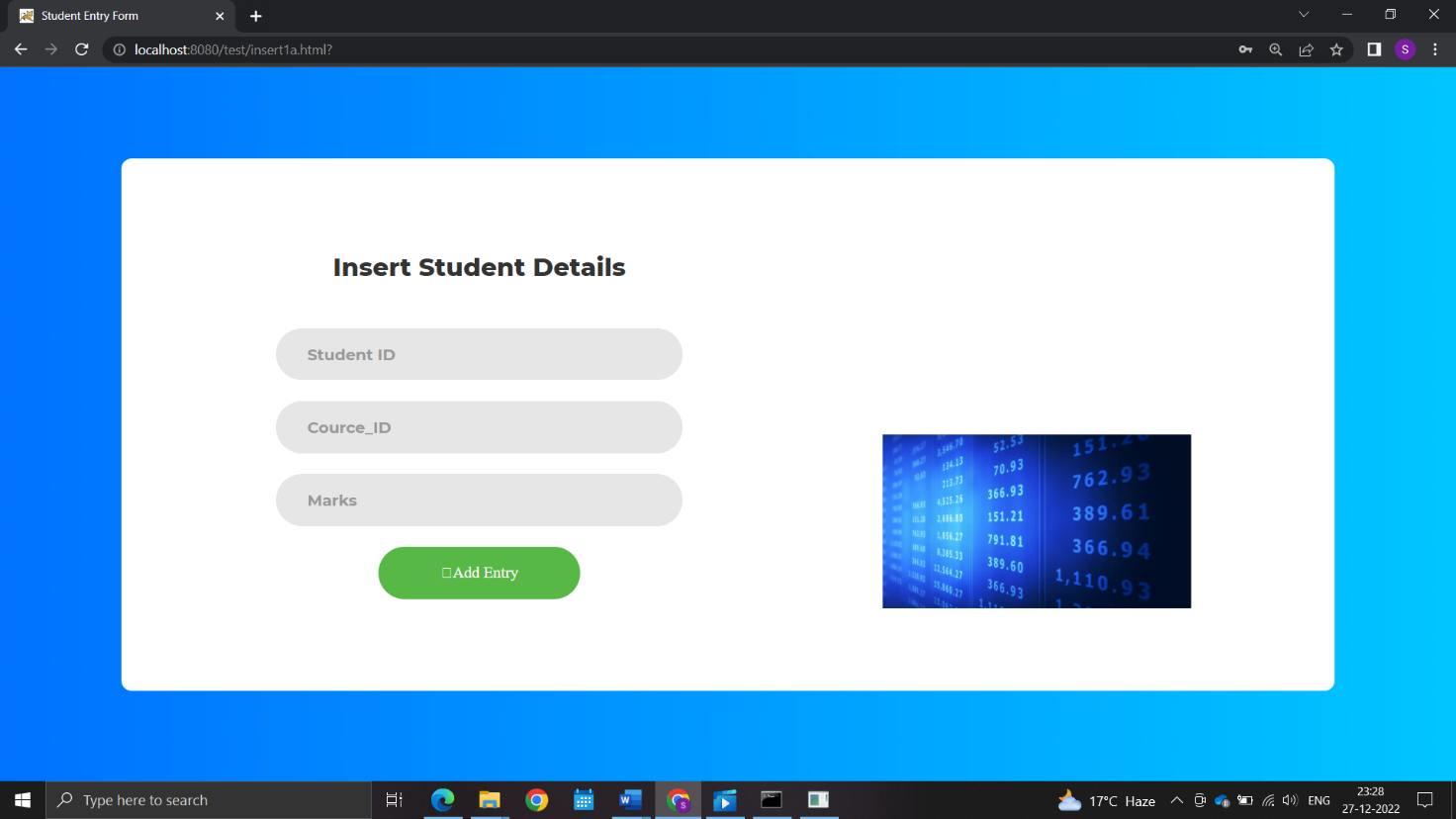




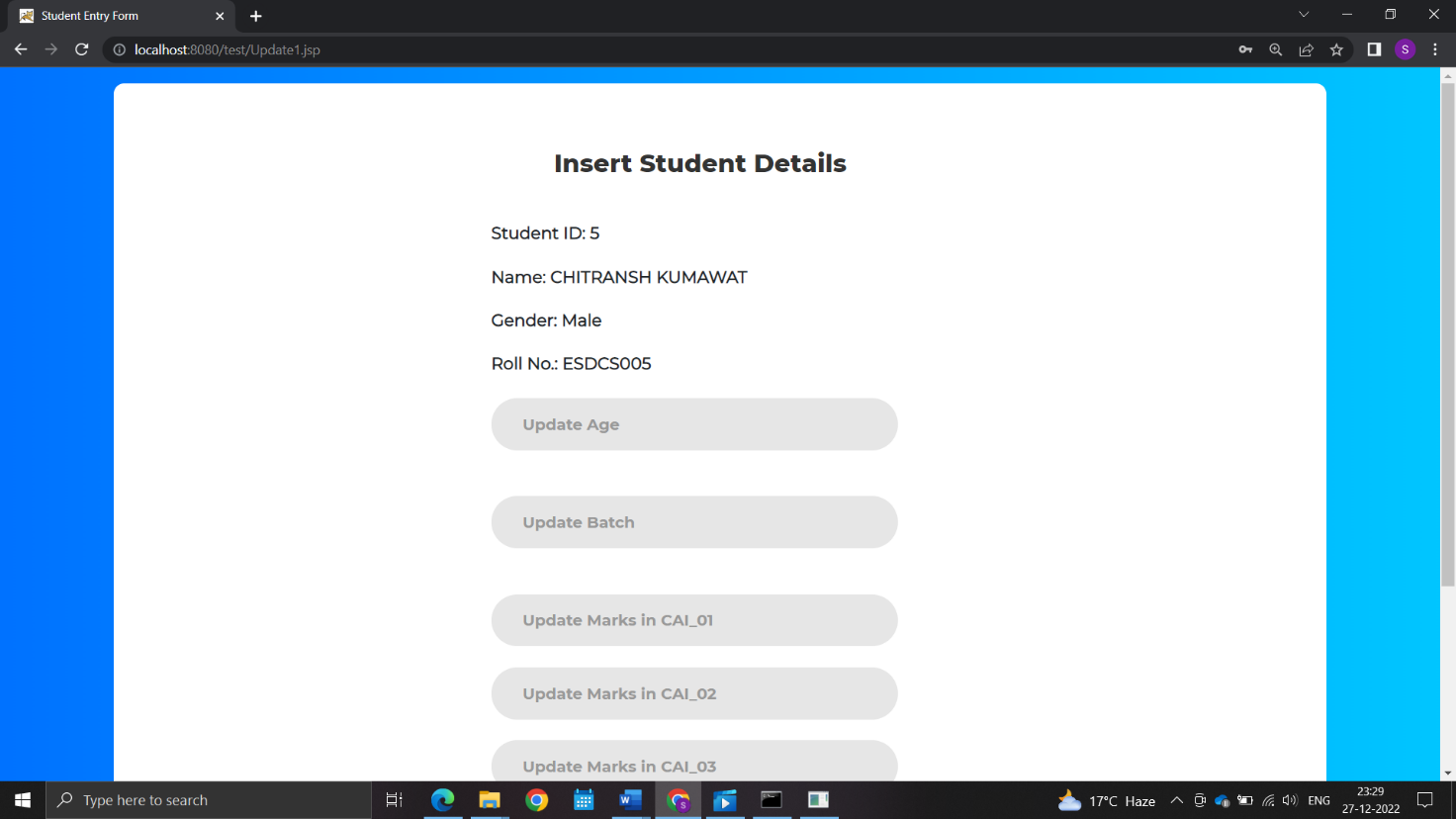


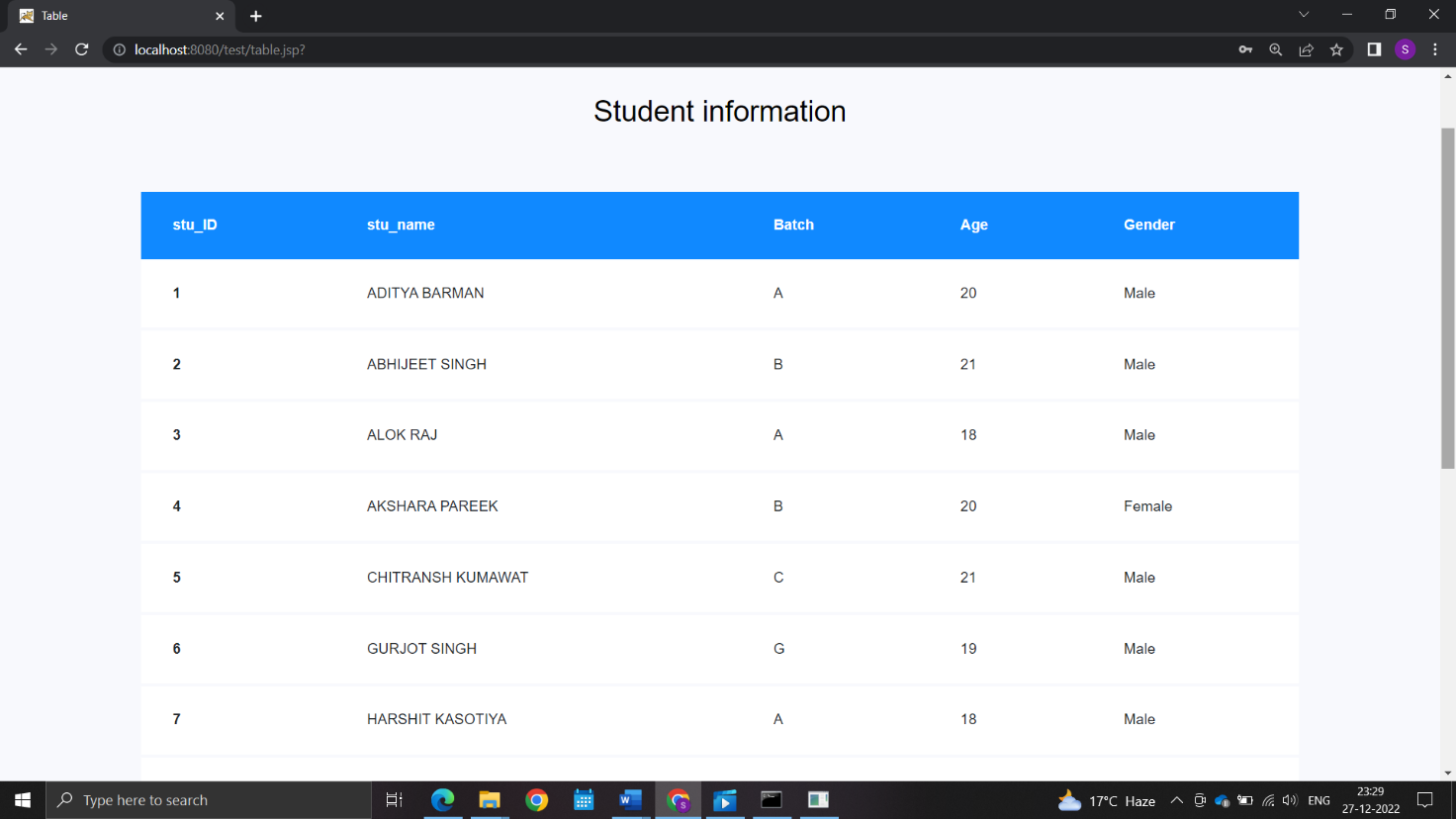
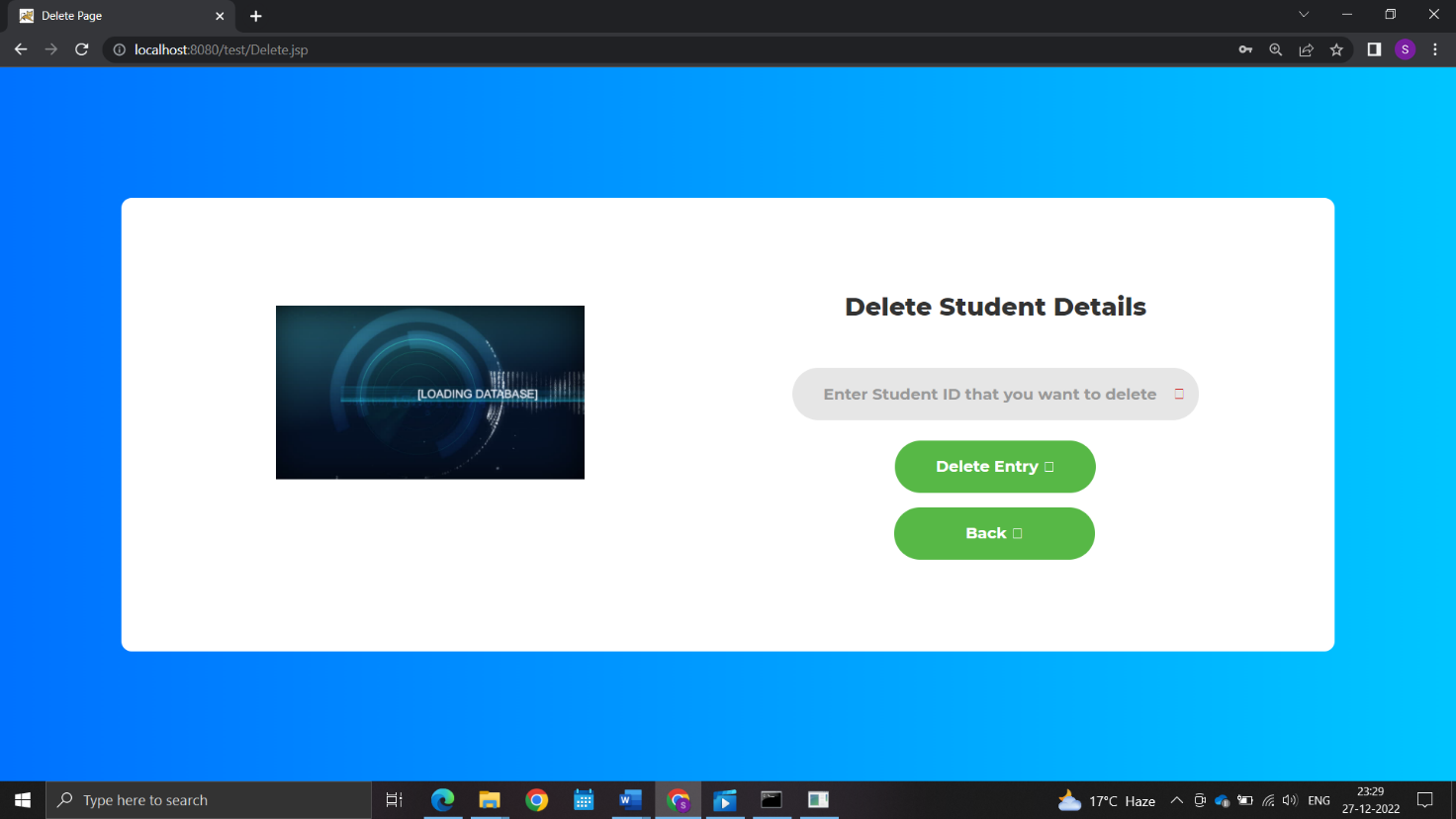
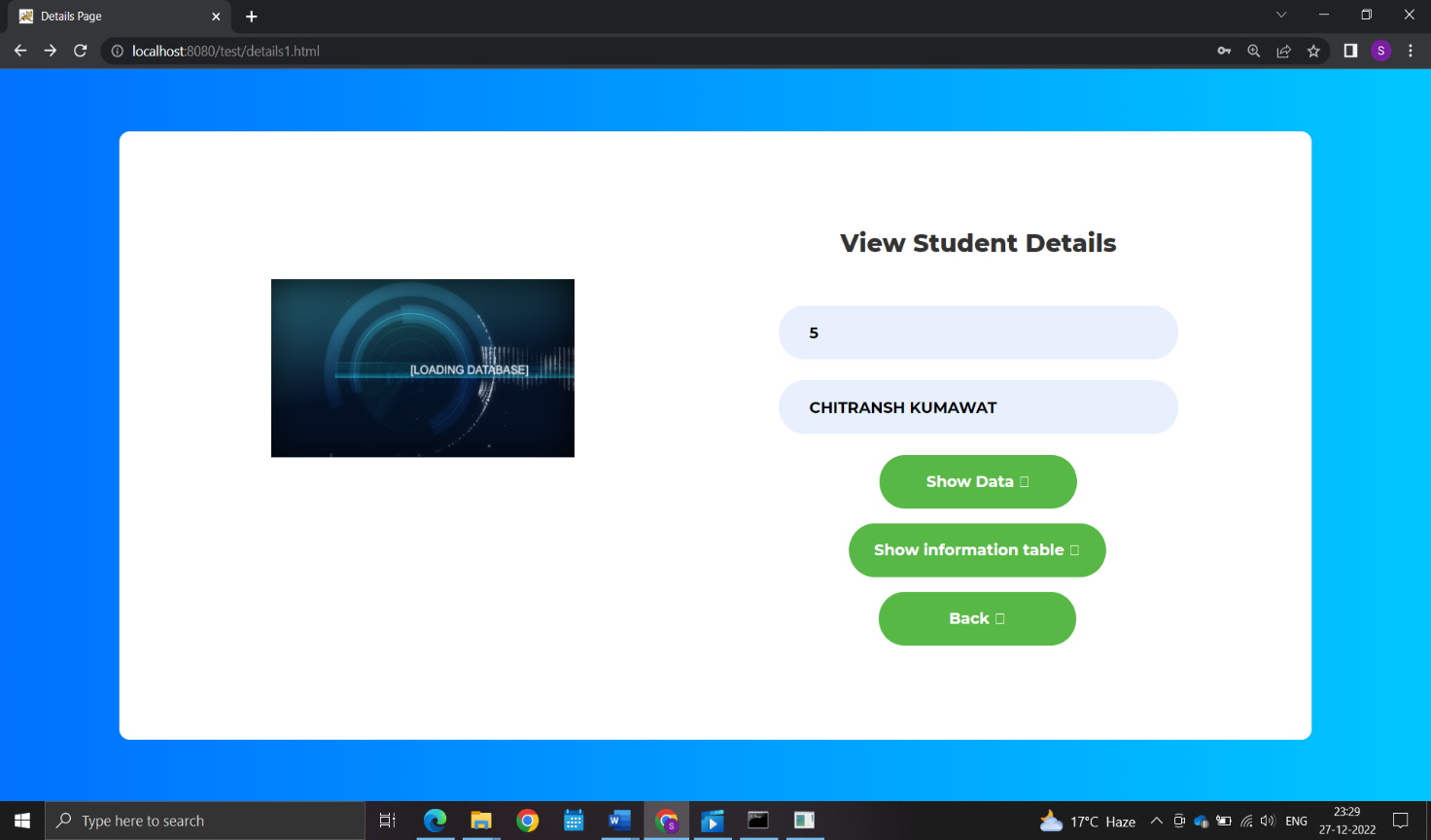


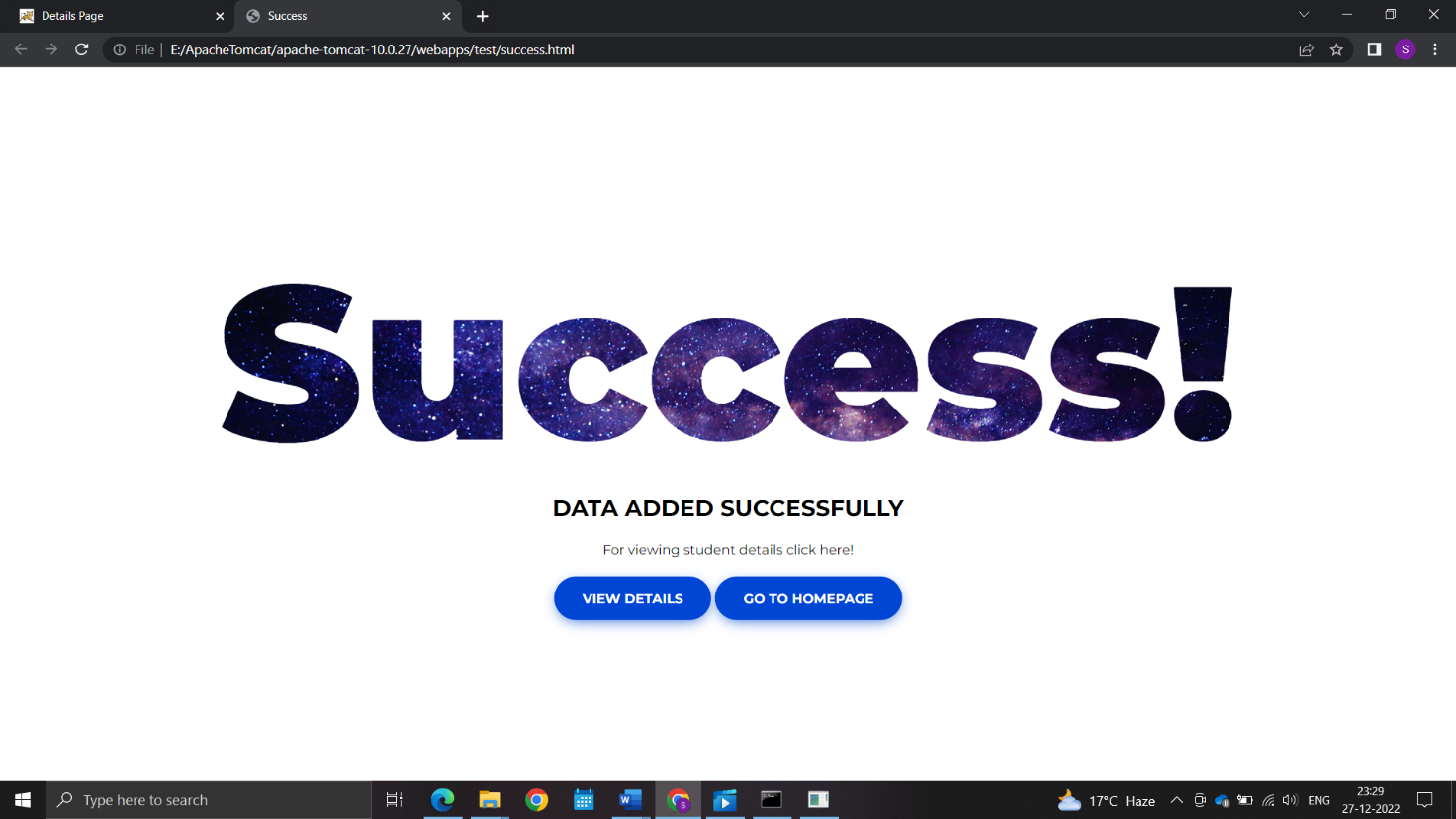
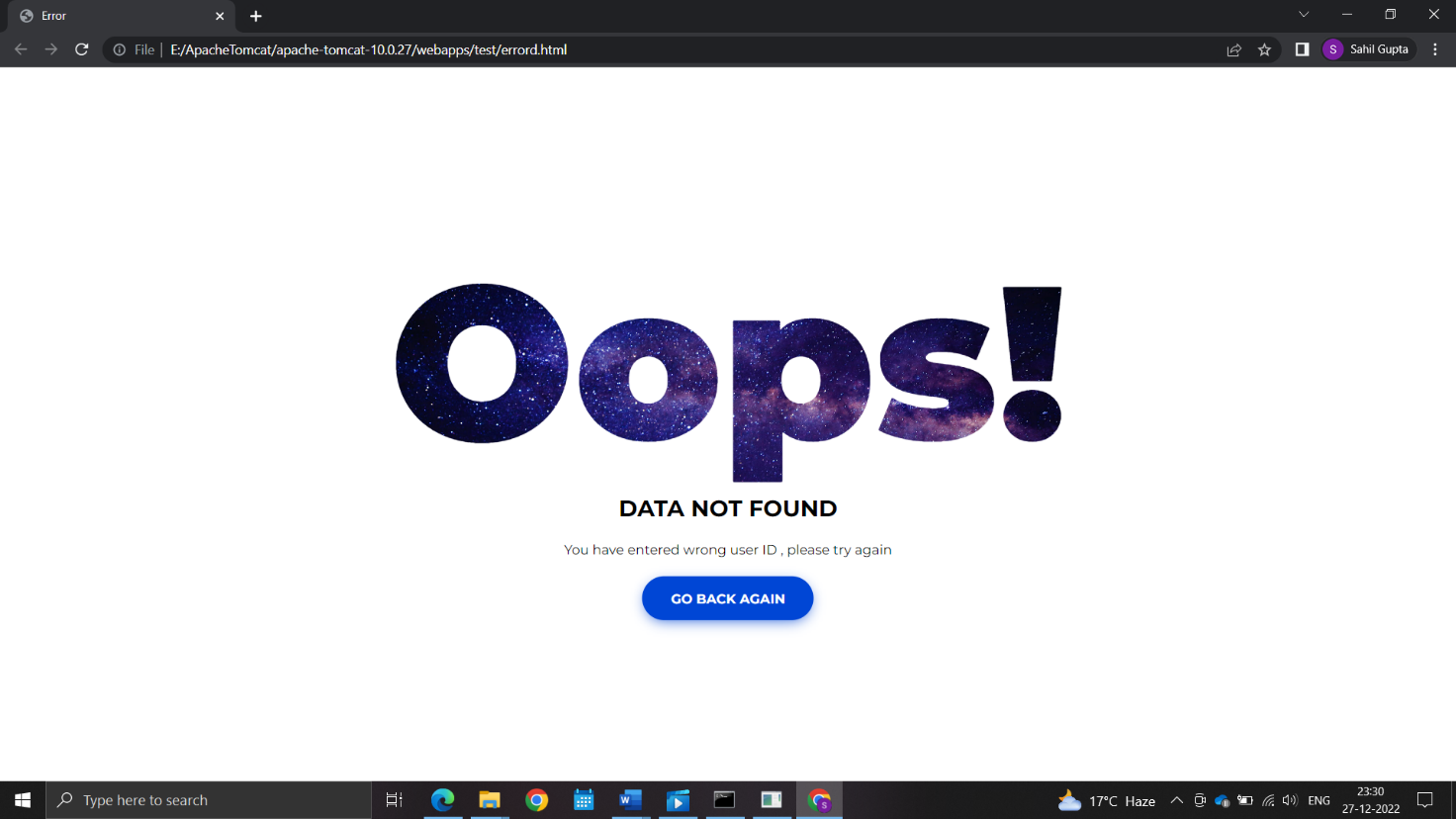
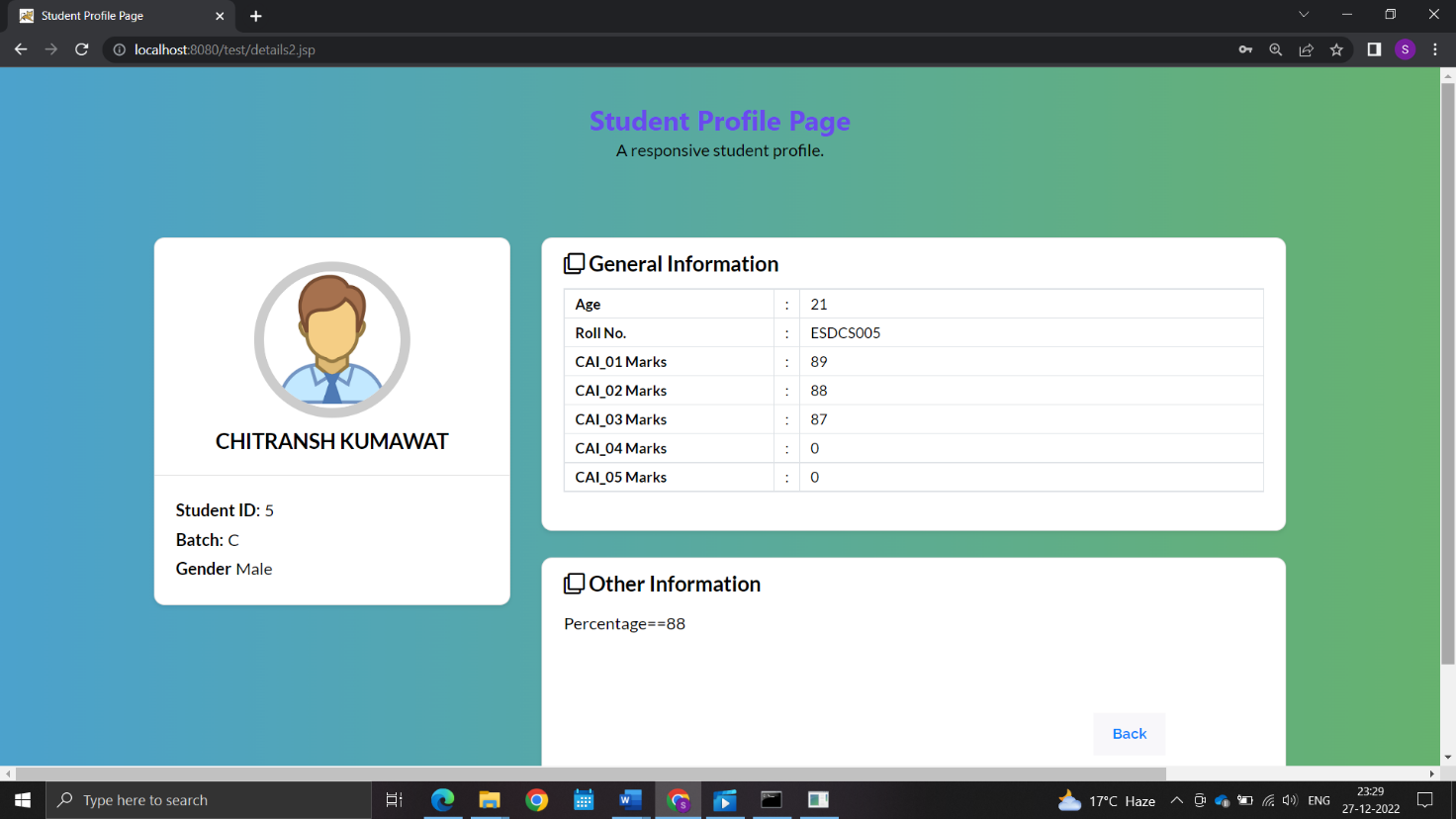






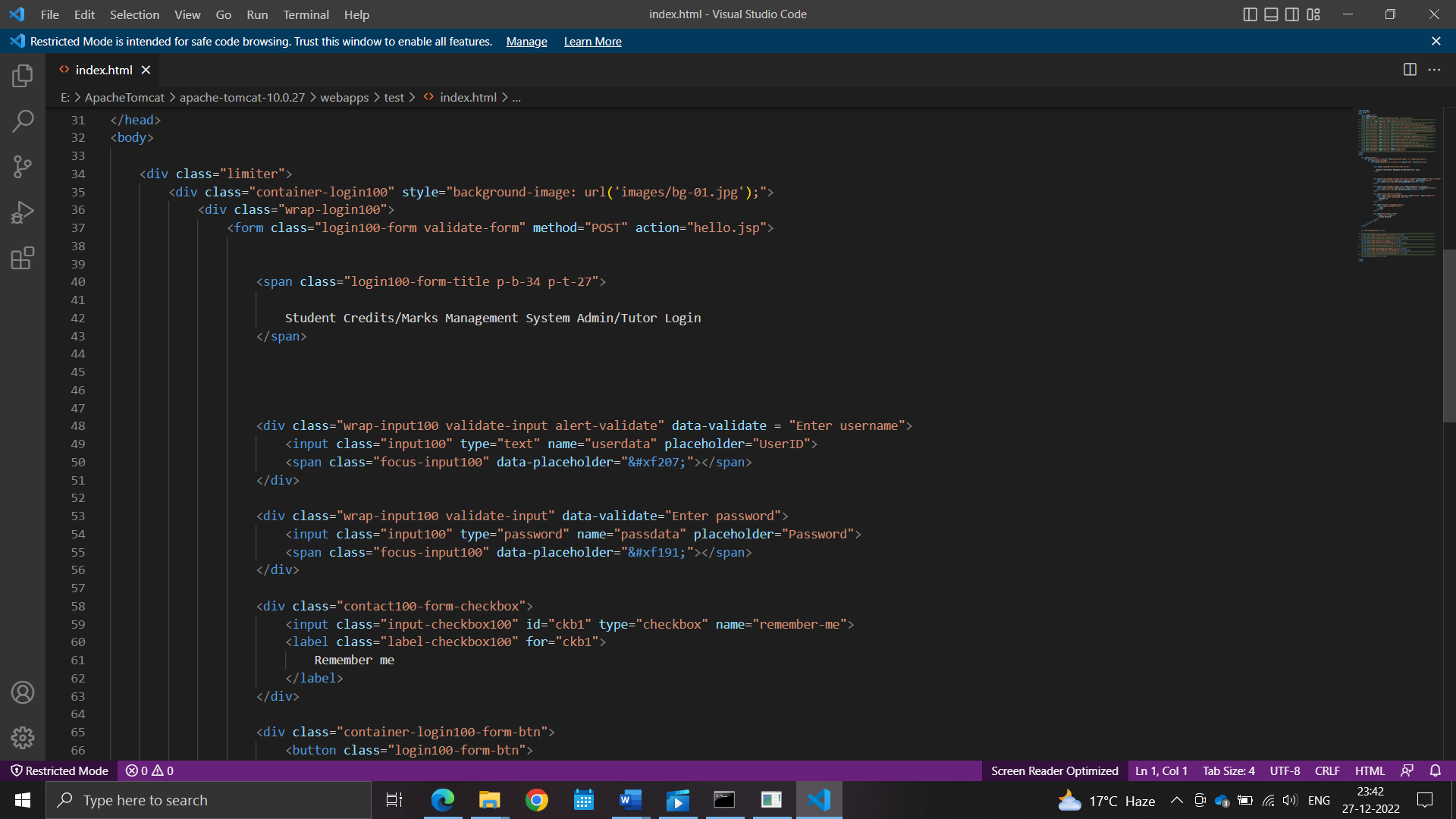


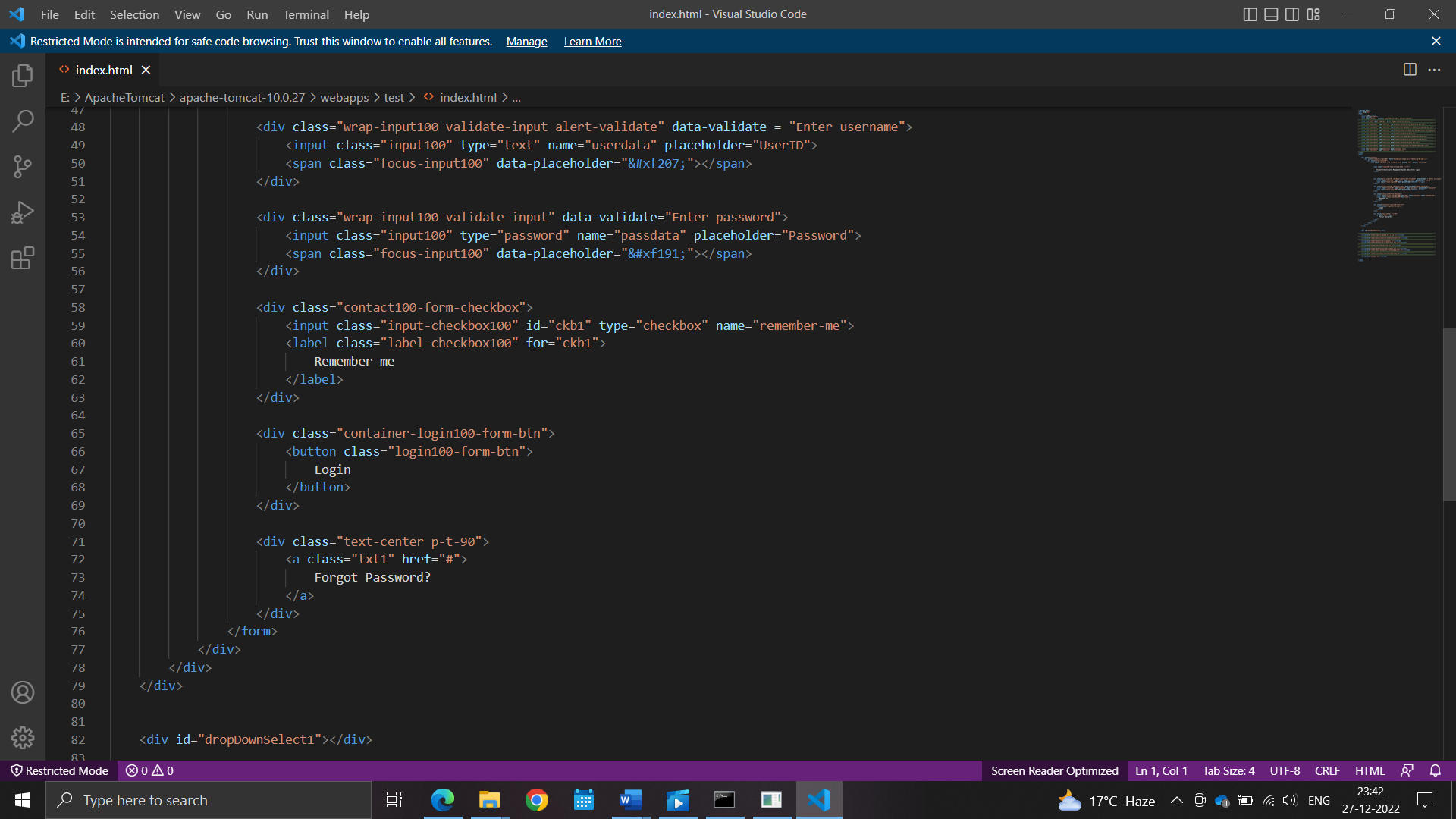


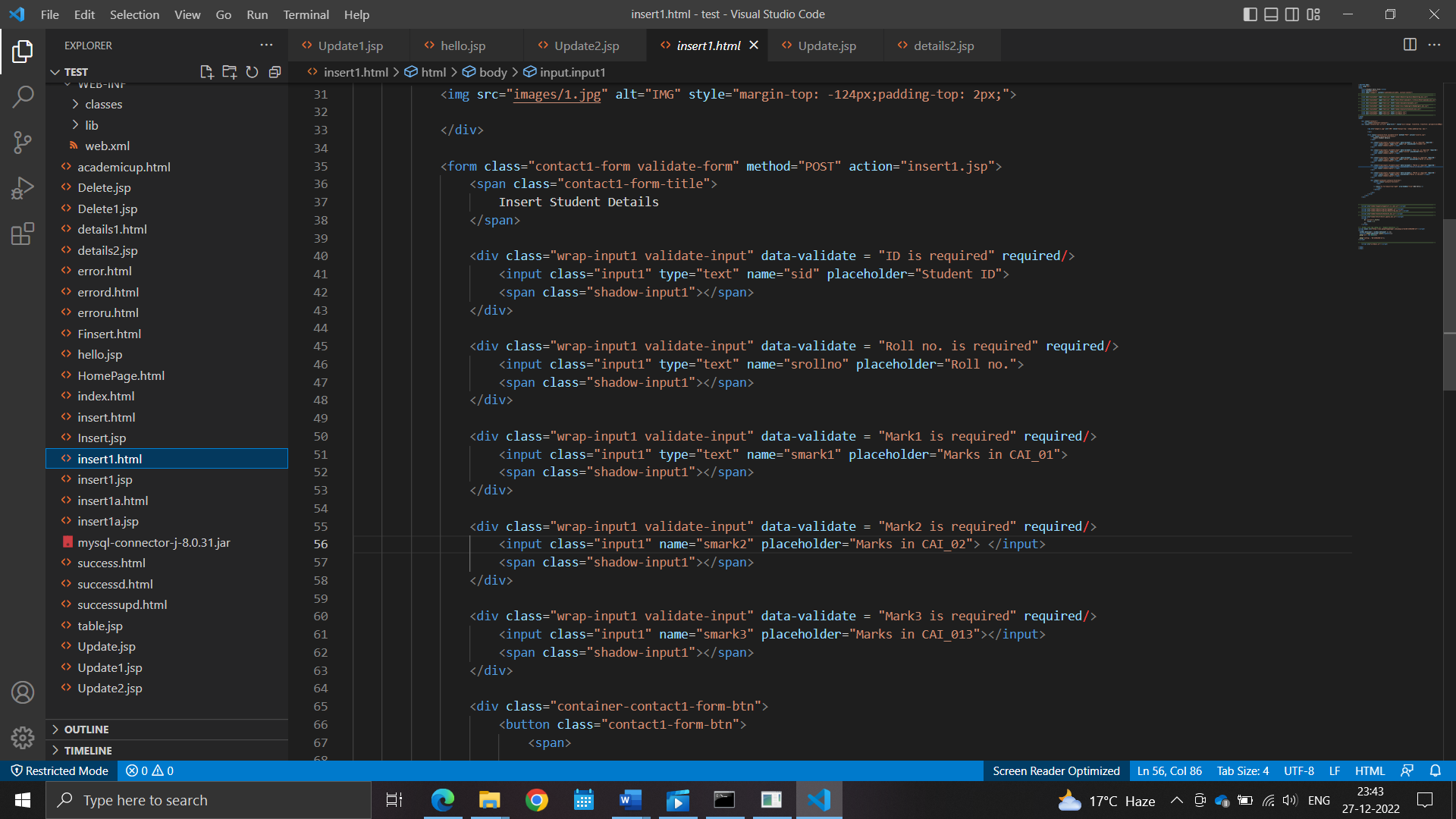
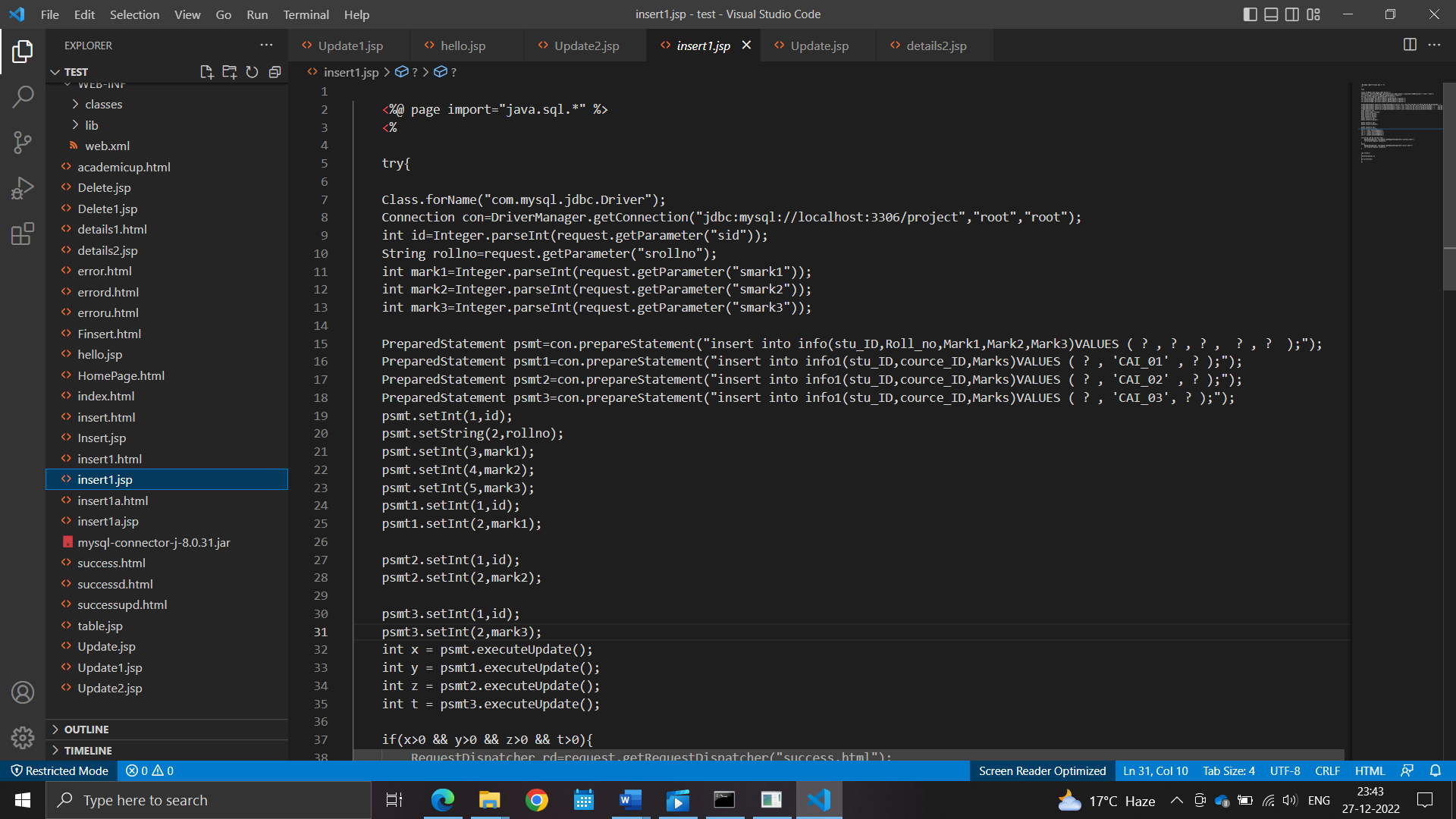
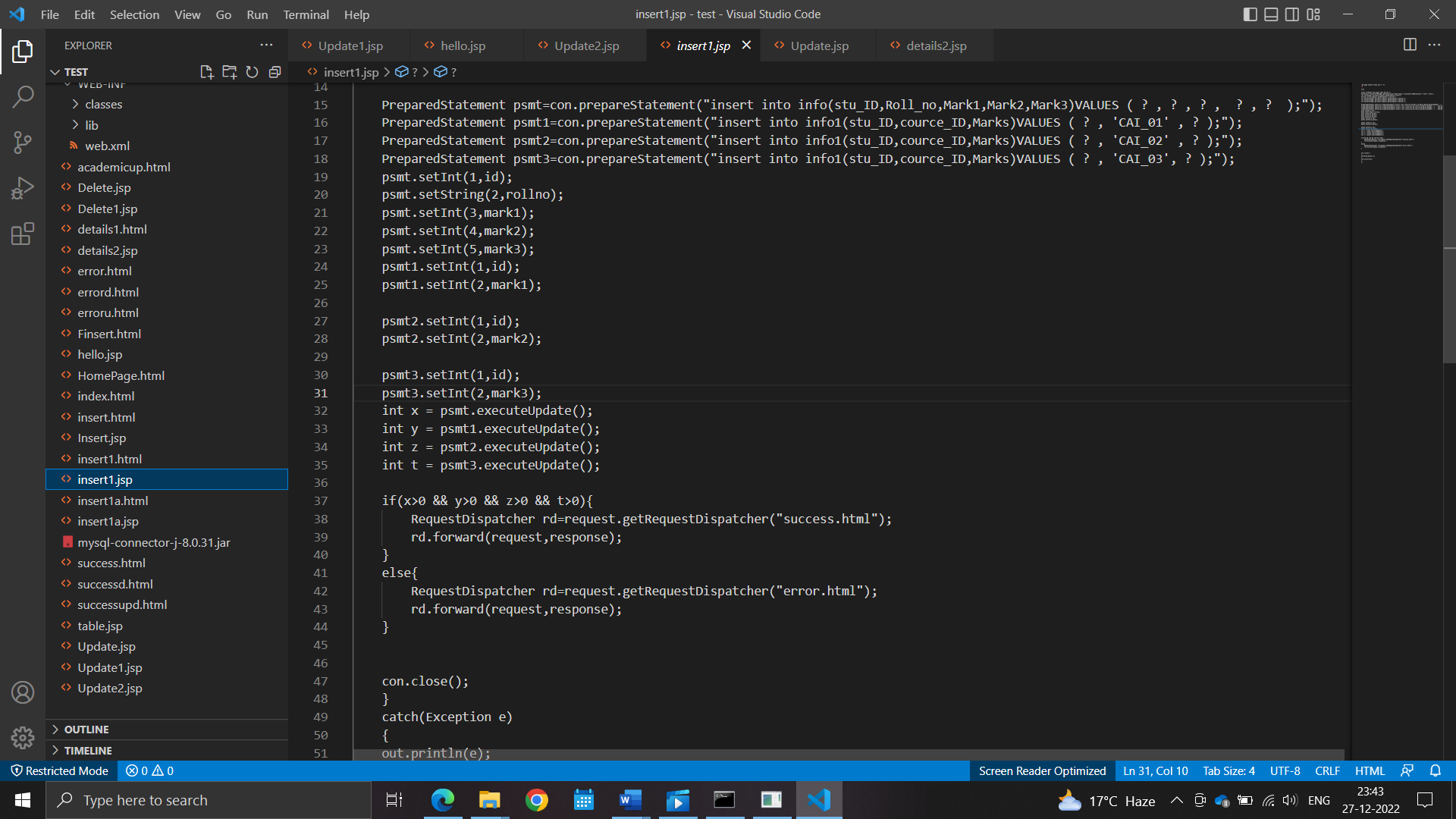
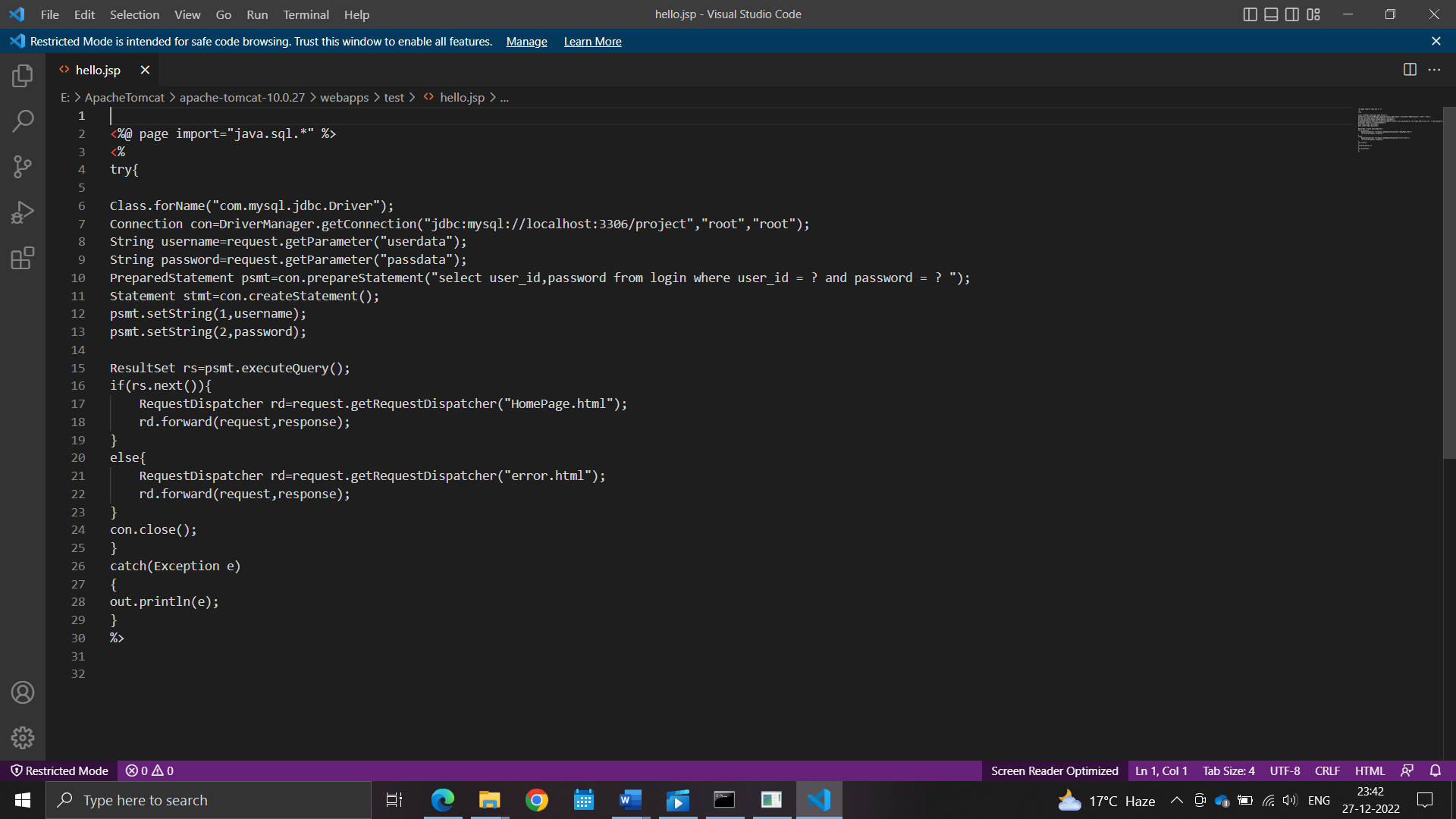
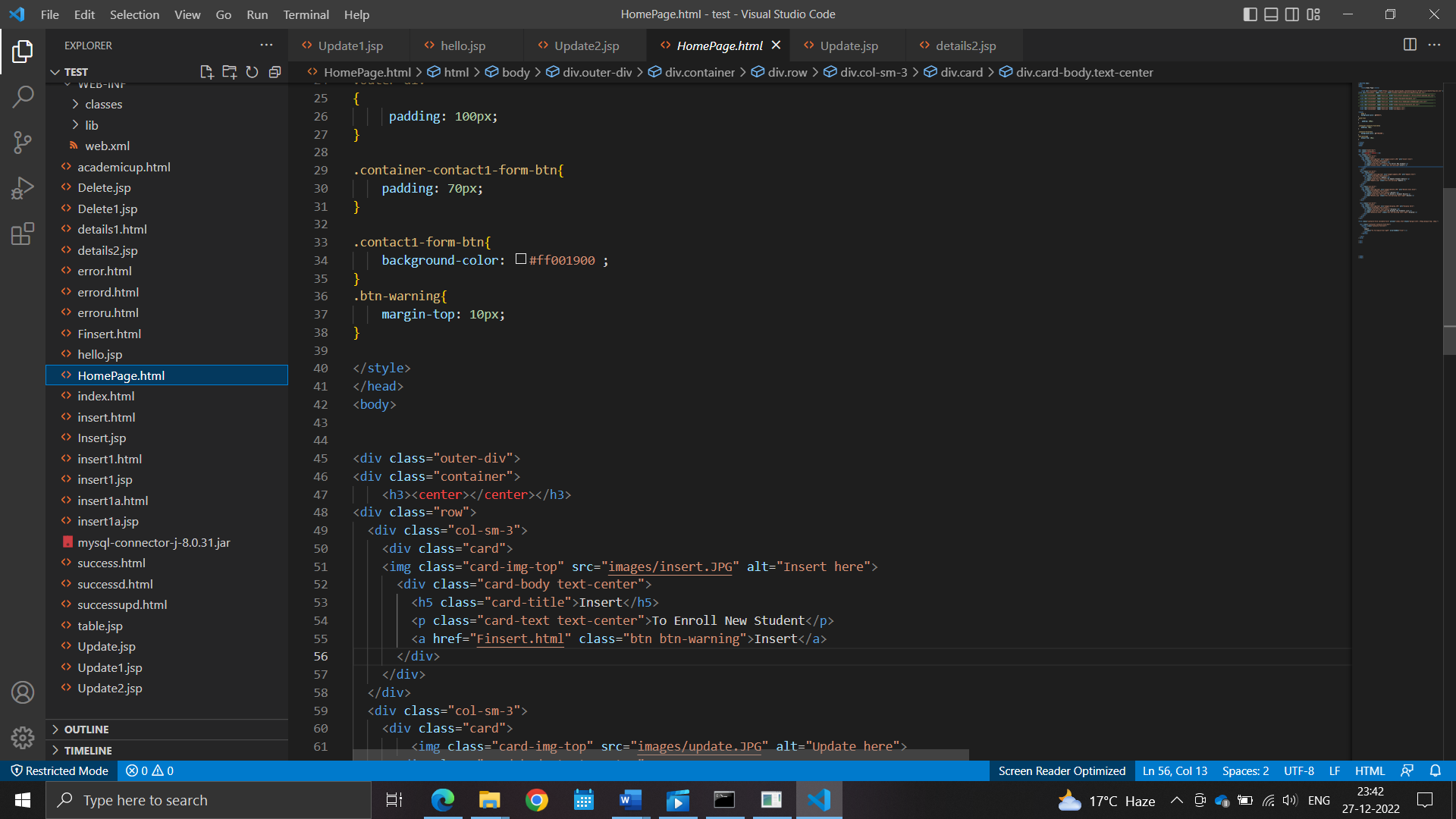
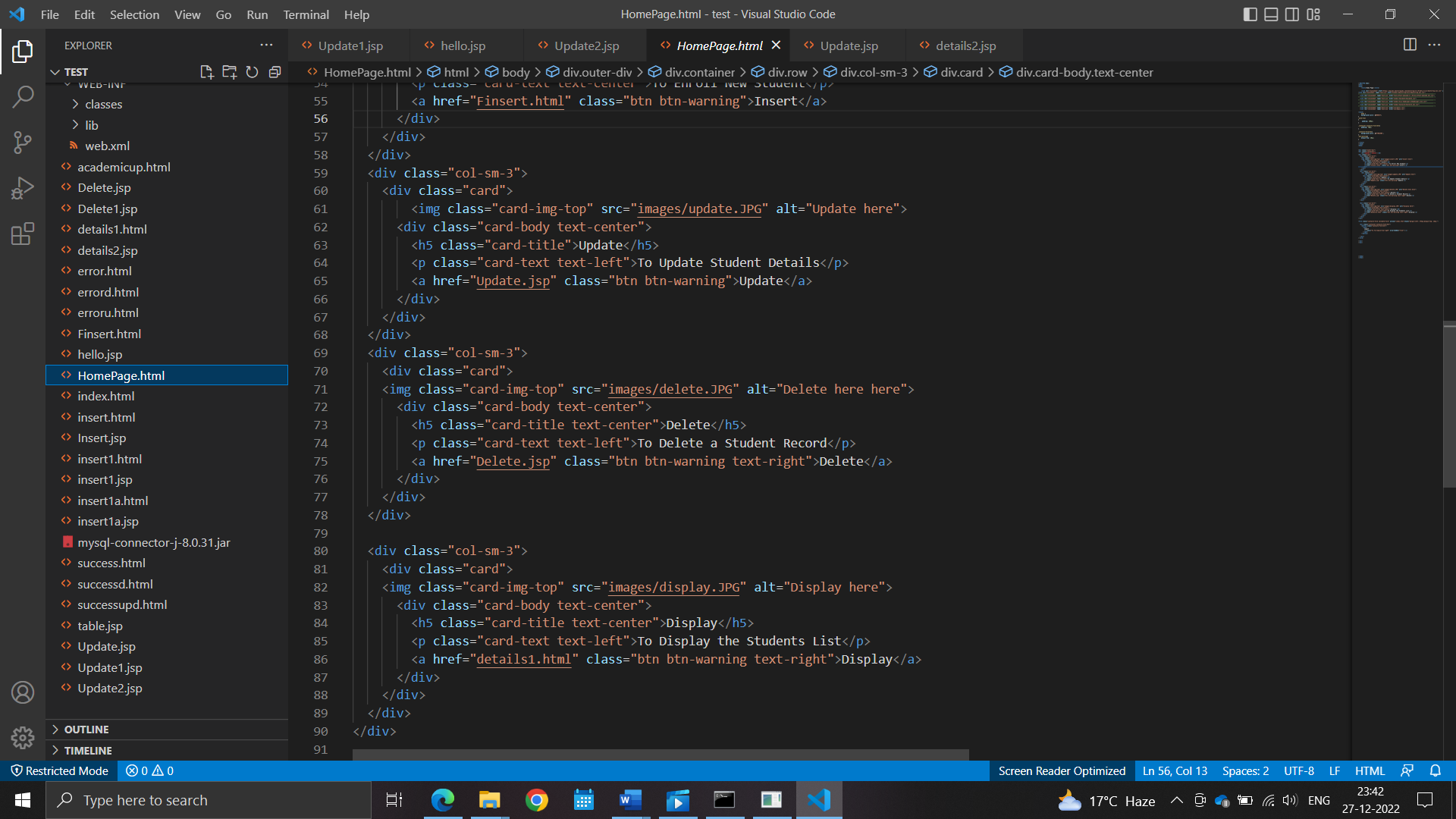


**Chapter 7**

**CODE OF PROJECT**







**For more details of the code.**

* **View at: --** https://github.com/guptasahil5635/NSP.git