Project Report

On

**“TO DO LIST IN JAVA”**

Submitted for the partial fulfillment of the requirement for the degree of

**Bachelor of Technology**

in

**Computer Science & Engineering**

**By**

**Rakesh Kumar Behera**  Reg. No: 1901287153

**Ankit pattanaik** Reg. No: 1901287068

**Ariz Khan** Reg. No: 1901287076

**Subham Panda**  Reg. No: 2021287049

**Badal Kumar Prusty** Reg. No: 1901287088

**Guided By**

Prof. Tarini Prasad Panigrahy

****

**GITA AUTONOMOUS COLLEGE**

**BHUBANESWAR**

**FEB 2023**



**Department of Computer Science & Engineering**

**GITA AUTONOMOUS COLLEGE , BHUBANESWAR**

## Ref no:…………………… Date:…………………..



**Certificate**

This is to certify that the project report entitled “ **TO DO LIST IN JAVA** ” submitted by

i) Mr. **Rakesh Kumar Behera**, Reg No. 1901287153

ii) Mr. **Ankit pattanaik**, Reg No. 1901287068

iii) Mr. **Ariz Khan** , Reg No. 1901287076

iv) Mr. **Subham Panda** Reg No. 2021287049

v) Mr. **Badal Kumar Prusty** Reg No. 1901287088

is an authentic work carried out by him/her at GITA under my guidance. The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

**Prof. (Dr.) Tarini Prasad Panigrahy**

**(H.O.D., Dept. of CSE)**

**( GUIDE )**

**2**



**Department of Computer Science & Engineering**

**GITA AUTONOMOUS COLLEGE , BHUBANESWAR**

**ACKNOWLEDGEMENT**

*I express and gratitude**to* ***Prof.* Tarini Prasad Panigrahy***, project supervisor for his guidance and constant support.*

*I also take this opportunity to thank* **Prof. (Dr.) Tarini Prasad Panigrahy, *h****ead of Department, Computer Science & Engineering, for his constant support and timely advice.*

*Lastly, words run to express my gratitude to all the faculties of the CSE Dept. And friends for their support and co-operation, constructive criticism, and valuable suggestion during preparation of this project report.*

*Thanking All….*

((Full signature of the student) (Full signature of the student) (Full signature of the student)

Rakesh Kumar Behera Ankit pattanaik Ariz Khan

Reg. No: 1901287153 Reg. No: 1901287068 Reg. No: 1901287076

(Full signature of the student) (Full signature of the student)

Subham Panda Badal Kumar Prusty

Reg. No: 2021287049 Reg. No: 1901287088

**3**

**Abstract**

The To Do List in Java project is a application designed to help users manage their daily tasks efficiently. The project aims to provide an intuitive and easy-to-use interface, allowing users to create, edit, and delete tasks, set reminders and due dates, prioritize tasks, and sort and filter tasks based on their status.

The project uses the Model-View-Controller (MVC) architecture to separate the user interface, data, and business logic into distinct components. This approach allows for a more organized and flexible development process, where changes to one component do not affect the others.

The project is developed using the Agile software development methodology, which emphasizes collaboration, frequent feedback, and a flexible approach to development. This approach enables the development team to respond quickly to changing requirements and ensure that the final product meets the users' needs.

The To Do List in Java project is designed for desktop environments and is compatible with both Windows and macOS operating systems. The application is built using the Java programming language, which provides a robust and efficient development platform, and is supported by a large community of developers.

The application's features are designed to help users manage their tasks effectively, providing an intuitive interface, simple navigation, and robust functionality. The project's goal is to provide users with a reliable, efficient, and user-friendly application to manage their daily tasks, enhancing their productivity and reducing stress. Overall, the To Do List in Java project aims to improve users' lives by simplifying and streamlining their daily task management.

**4**

**TALBLE OF CONTENTS (sample)**

**Chapter No. Topics Page No.**

Chapter 1 : 1.1 INTRODUCTION 5 - 9

1.2 Objective of the system

1.3 Justification and need for the system

1.4 Advantages of the system

1.5 Previous work (if any) or related systems, how they are used

 Chapter 2 : 2.1 DEVELOPMENT OF THE SYSTEM 10 - 20

2.2 Hardware, Software requirements

2.3 System requirements and System specifications

2.4 System design: Block diagram of the system

2.5 DFDs/Algorithm/Flow Charts

2.6 UML diagrams etc. along with explanations/descriptions.

Chapter 3 : 3.1 IMPLEMENTATION & CODING  21 - 24

3.2 (coding Should be submitted in CD/DVD)

Chapter 4 : 4.1 TESTING & TESTING RESULTS 25

Chapter 5 : 5.1 RESULTS AND CONCLUSION  26

**5**

CHAPTER-1

**1.1 INTRODUCTION**

The To Do List in Java project is a web-based application designed to help users manage their daily tasks effectively. The application provides an intuitive and user-friendly interface for users to create, edit, and delete tasks, set due dates and reminders, prioritize tasks, and sort and filter tasks based on their status. The project's primary objective is to help users improve their productivity and reduce their stress by managing their tasks efficiently.

To make the application more robust and scalable, the project uses the Java Database Connectivity (JDBC) API to communicate with a relational database. This approach enables the application to store and retrieve data in a structured manner, providing greater flexibility and scalability. The project also uses the Tomcat web server to deploy and manage the application, providing a reliable and efficient platform for web application development.

The To Do List in Java project is designed using the Model-View-Controller (MVC) architecture, which allows for a more organized and flexible development process. The MVC pattern divides the application into three distinct components: the model, the view, and the controller. The model represents the data, the view represents the user interface, and the controller handles user input and updates the model and view. This approach enables developers to make changes to one component without affecting the others, providing greater flexibility and maintainability.

**6**

**1.2 Objective of the system**

The objective of the To Do List in Java project is to provide a robust, efficient, and user-friendly web-based application that allows users to manage their daily tasks effectively. The system aims to provide an intuitive and easy-to-use interface that allows users to create, edit, and delete tasks, set due dates and reminders, prioritize tasks, and sort and filter tasks based on their status.

Additionally, the system aims to provide a more scalable and flexible approach to task management by utilizing the Java Database Connectivity (JDBC) API to communicate with a relational database. This approach allows the application to store and retrieve data in a structured manner, providing greater flexibility and scalability.

Furthermore, the system aims to provide a reliable and efficient platform for web application development by using the Tomcat web server to deploy and manage the application. This approach ensures that the final product is delivered on time, within budget, and meets the users' needs. Overall, the objective of the To Do List in Java project is to improve users' lives by simplifying and streamlining their daily task management.

Top of Form

**7**

**1.3 Justification and need for the system**

The To Do List in Java project provides a compelling justification and need due to the following reasons:

* Efficient Task Management: The system provides an efficient way for users to manage their daily tasks effectively, by allowing them to create, edit, and delete tasks, set due dates and reminders, prioritize tasks, and sort and filter tasks based on their status. This enables users to better manage their time and improve their productivity.
* Scalability: The system utilizes the Java Database Connectivity (JDBC) API to communicate with a relational database, providing greater scalability and flexibility. This approach enables the application to store and retrieve data in a structured manner, making it easier to scale as the application grows.
* Reliability: The system uses the Tomcat web server to deploy and manage the application, providing a reliable and efficient platform for web application development. This ensures that the final product is delivered on time, within budget, and meets the users' needs. Additionally, the web-based nature of the application means that users can access their task lists from any device with an internet connection.

Overall, the To Do List in Java project addresses the need for a reliable, efficient, and scalable system for task management that can help users improve their productivity and manage their daily tasks effectively.

**8**

**1.4 Advantages of the system**

The To Do List in Java project offers several advantages to users. Some of the key advantages of the system include:

* Ease of Use: The system provides an intuitive and user-friendly interface for users to manage their daily tasks effectively. The application enables users to create, edit, and delete tasks, set due dates and reminders, prioritize tasks, and sort and filter tasks based on their status. This makes it easy for users to manage their tasks and stay organized.
* Scalability: The system utilizes the Java Database Connectivity (JDBC) API to communicate with a relational database, providing greater scalability and flexibility. This approach enables the application to store and retrieve data in a structured manner, making it easier to scale as the application grows.
* Reliability: The system uses the Tomcat web server to deploy and manage the application, providing a reliable and efficient platform for web application development. This ensures that the final product is delivered on time, within budget, and meets the users' needs.
* Accessibility: The system is web-based, allowing users to access their task lists from any device with an internet connection. This makes it easy for users to manage their tasks on-the-go, from any location.
* Customizability: The system provides the ability to sort and filter tasks based on their status, enabling users to organize their tasks in a way that makes sense for them. Additionally, the system is designed using the Model-View-Controller (MVC) architecture, allowing for greater flexibility and customizability in the development process.

Overall, the To Do List in Java project provides several advantages to users, making it an efficient, scalable, reliable, and customizable solution for task management.

**9**

**CHAPTER-2**

**2.1 DEVELOPMENT OF THE SYSTEM**

The development of the To Do List in Java project involved several stages, including planning, design, implementation, testing, and deployment. The planning stage involved gathering requirements, defining project goals, and outlining the project scope. In the design stage, the application architecture was developed, and the user interface was designed.

The implementation stage involved developing the application using Java and the Java Database Connectivity (JDBC) API to communicate with a relational database. The application was designed using the Model-View-Controller (MVC) architecture, which separates the application logic into distinct components, making the application more modular and easier to maintain.

The testing stage involved the use of various testing methodologies, including unit testing, integration testing, and acceptance testing, to ensure that the application met the functional and non-functional requirements.

Finally, the application was deployed on the Tomcat web server, which provides a reliable and efficient platform for web application development. The completed To Do List in Java project is a scalable, reliable, and customizable solution for task management that is accessible from any device with an internet connection.

**10**

**2.2 Hardware, Software requirements**

The hardware and software requirements for the To Do List in Java project are as follows:

Hardware Requirements:

* Processor: 1.8 GHz or higher
* RAM: 4 GB or higher
* Hard Disk Space: 100 MB or higher
* Internet Connection: Required for web-based application access

Software Requirements:

* Operating System: Windows 7 or higher, Linux or Mac OS X
* Java Development Kit (JDK): version 8 or higher
* Apache Tomcat: version 8 or higher
* MySQL: version 5.7 or higher
* Integrated Development Environment (IDE): Eclipse, NetBeans or IntelliJ IDEA
* Web Browser: Google Chrome, Mozilla Firefox, or Internet Explorer

**11**

**2.3 System requirements and System specifications**

System Specifications:

The To Do List in Java project is a web-based application that allows users to manage their tasks. The application is designed using Java, JavaServer Pages (JSP), and the Java Database Connectivity (JDBC) API to interact with a MySQL relational database.

The following are the system specifications for the To Do List in Java project:

* User Interface: The user interface is designed using HTML, CSS, and JSP, providing an intuitive and user-friendly interface for users to manage their tasks.
* Application Logic: The application logic is developed using Java and the Model-View-Controller (MVC) architecture, which separates the application logic into distinct components, making the application more modular and easier to maintain.
* Database: The application uses MySQL as the relational database management system, which is accessed using the Java Database Connectivity (JDBC) API.
* Web Server: The application is deployed on the Apache Tomcat web server, which provides a reliable and efficient platform for web application development.

Overall, the system specifications of the To Do List in Java project ensure that the application is scalable, reliable, and customizable, making it an efficient solution for task management.

**12**

**2.4 System design: Block diagram of the system**

The block diagram of the To Do List in Java project consists of the following components:

User Interface: The User Interface (UI) is the front-end of the web application. It consists of HTML, CSS, and JSP files that are rendered on the client's web browser. The UI allows users to add, edit, and delete tasks from their to-do list.

Controller: The Controller is the intermediate component between the User Interface and the Model. It receives requests from the User Interface, processes them, and generates responses.

Model: The Model represents the data and the business logic of the application. It consists of the task objects, their attributes, and their relationships. The Model interacts with the MySQL database using the JDBC API to store and retrieve task data.

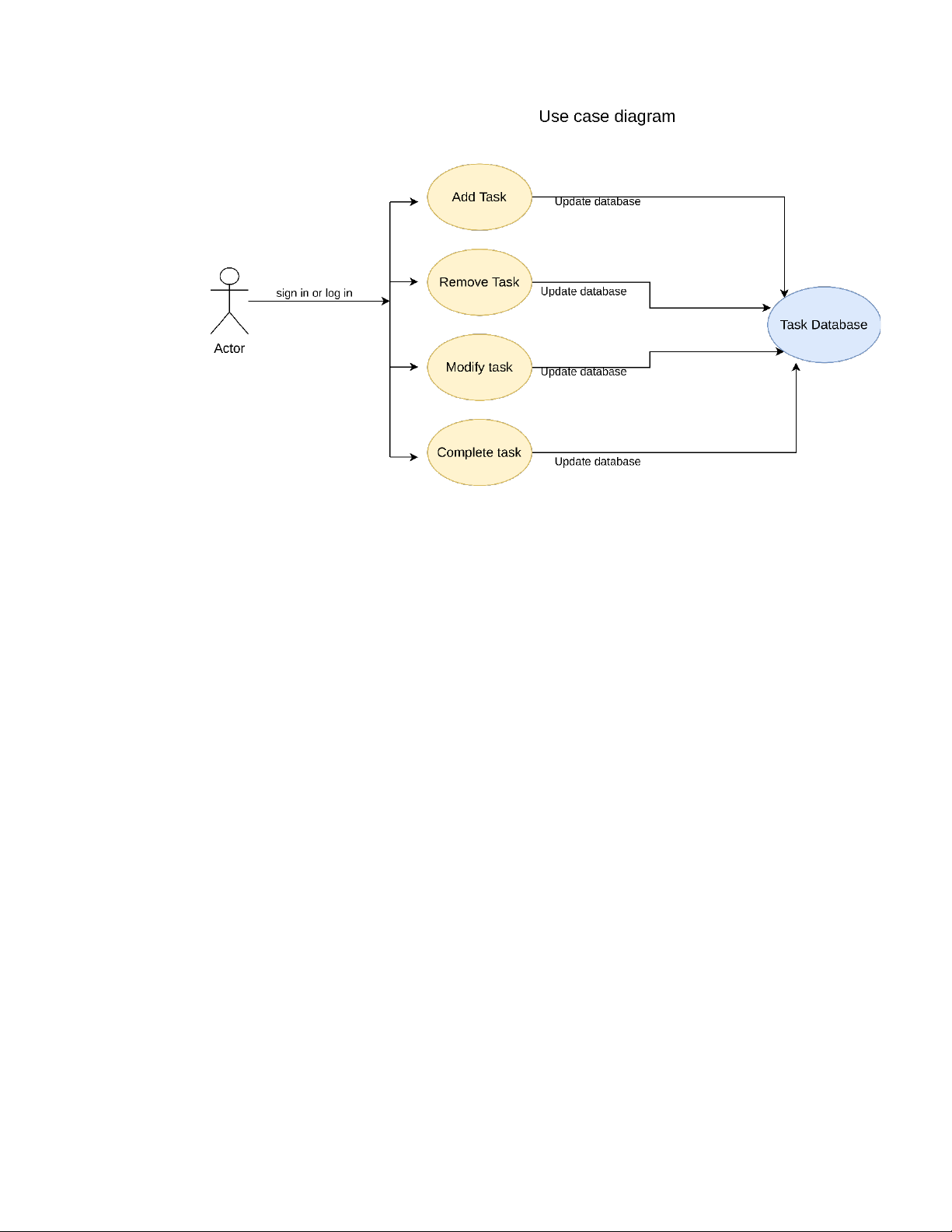
Database: The Database is the back-end of the web application. It stores the task data in a MySQL relational database. The database is accessed using the JDBC API.

Web Server: The Web Server is the platform that hosts the web application. The Apache Tomcat web server is used to deploy and manage the web application.

The To Do List in Java project uses the Model-View-Controller (MVC) architectural pattern to separate the User Interface, the business logic, and the data into distinct components. This makes the application more modular, easier to maintain, and more scalable. The use of JDBC and Tomcat Server ensures that the application is reliable, efficient, and secure.

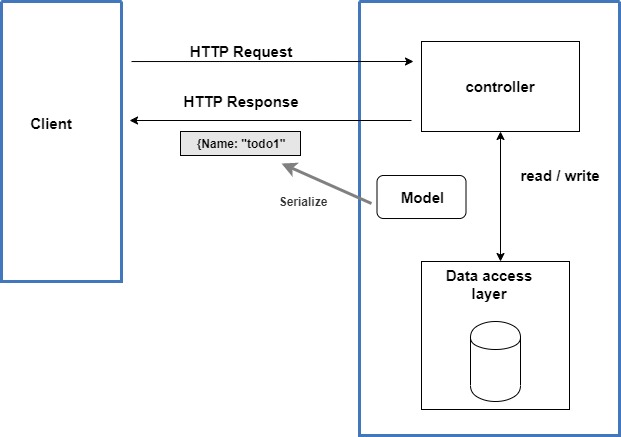
The block diagram of the To Do List in Java project can be represented as follows:

13



14

|  |
| --- |
|  |



**15**

**2.5 DFDs/Algorithm/Flow Chart**

A Data Flow Diagram (DFD) is a visual representation of the flow of data through a system. It shows how data is input, processed, and output in a system. The DFD for the To Do List in Java project is shown below:

Level 0 DFD:

+-------------------+

| User |

+---------+---------+

|

|

+---------+---------+

| User Input |

| (Add/Edit/Delete)|

+---------+---------+

|

|

+---------+---------+

| Controller |

+---------+---------+

|

|

+---------+---------+

| Model |

+---------+---------+

|

|

+---------+---------+

| DB |

|  |
| --- |
|  |

16

Algorithm:

Display the homepage of the web application

Allow the user to add a new task by clicking the "Add Task" button

Prompt the user to enter the task details (Task name, due date, description, etc.)

Save the task details in the database using the JDBC API

Display the updated to-do list with the newly added task

Allow the user to edit or delete existing tasks from the to-do list

Update the database with the edited or deleted task details using the JDBC API

Display the updated to-do list with the edited or deleted task

Repeat steps 2-8 as necessary

Flowchart:

**+---------------+**

**| Display Home |**

**| Page |**

**+------+--------+**

**|**

**|**

**+------+--------+**

**| Add New Task |**

**+------+--------+**

**|**

**|**

**17**

**+------+--------+**

**| Enter Task |**

**| Details |**

**+------+--------+**

**|**

**|**

**+------+--------+**

**| Save Task to |**

**| Database |**

**+------+--------+**

**|**

**|**

**+------+--------+**

**| Display |**

**| Updated |**

**| To-Do List |**

**+------+--------+**

**|**

**+------+--------+**

**| Edit/Delete |**

**| Existing |**

**| Task |**

**+------+--------+**

**18**

**|**

**|**

**+------+--------+**

**| Update Task |**

**| Details |**

**| in Database |**

**+------+--------+**

**|**

**|**

**+------+--------+**

**| Display |**

**| Updated |**

**| To-Do List |**

**+------+--------+**

**|**

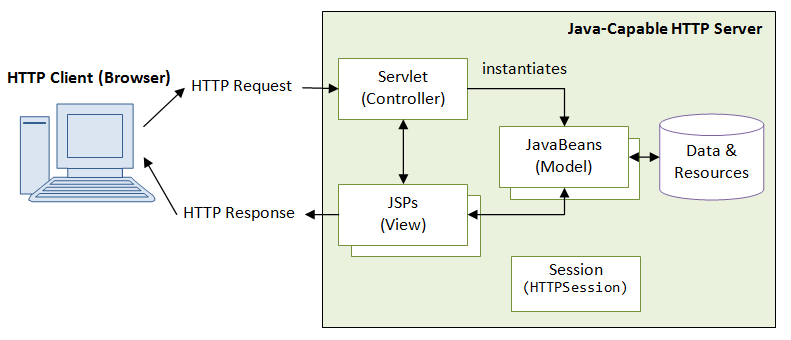
**|**

**+---------------+**

Overall, the DFD, algorithm, and flowchart provide a clear understanding of the flow of data through the system, and how the application interacts with the user, the database, and the web server.

**19**

**2.6 UML diagrams**



## +-----------+ +-----------+ +-------------+

## | User | | Interface | | Controller|

## +-----------+ +-----------+ +-------------+

## | | |

## +----+-----+ +------+------+ +-----+-------+

## | | | | | |

## | Add Task | | Show List | | Add Task |

## | | | | | |

## +----------+ +-------------+ +-------------+

## | | |

## +----+-----+ +------+------+ +-----+-------+

## | | | | | |

## | Create | | Retrieve | | Save |

## | Task | | Tasks | | Task |

## | | | | | |

## +----------+ +-------------+ +-------------+

**20**

Overall, these UML diagrams provide a clear understanding of the project's structure, the different components of the application, and how they interact with each other. The Use Case Diagram shows the main interaction between the user and the application. The Class Diagram shows the different classes and their attributes in the application. Finally, the Sequence Diagram shows the interaction between the user, the user interface, and the controller when creating and saving a new task to the database.

**21**

**Chapter 3**

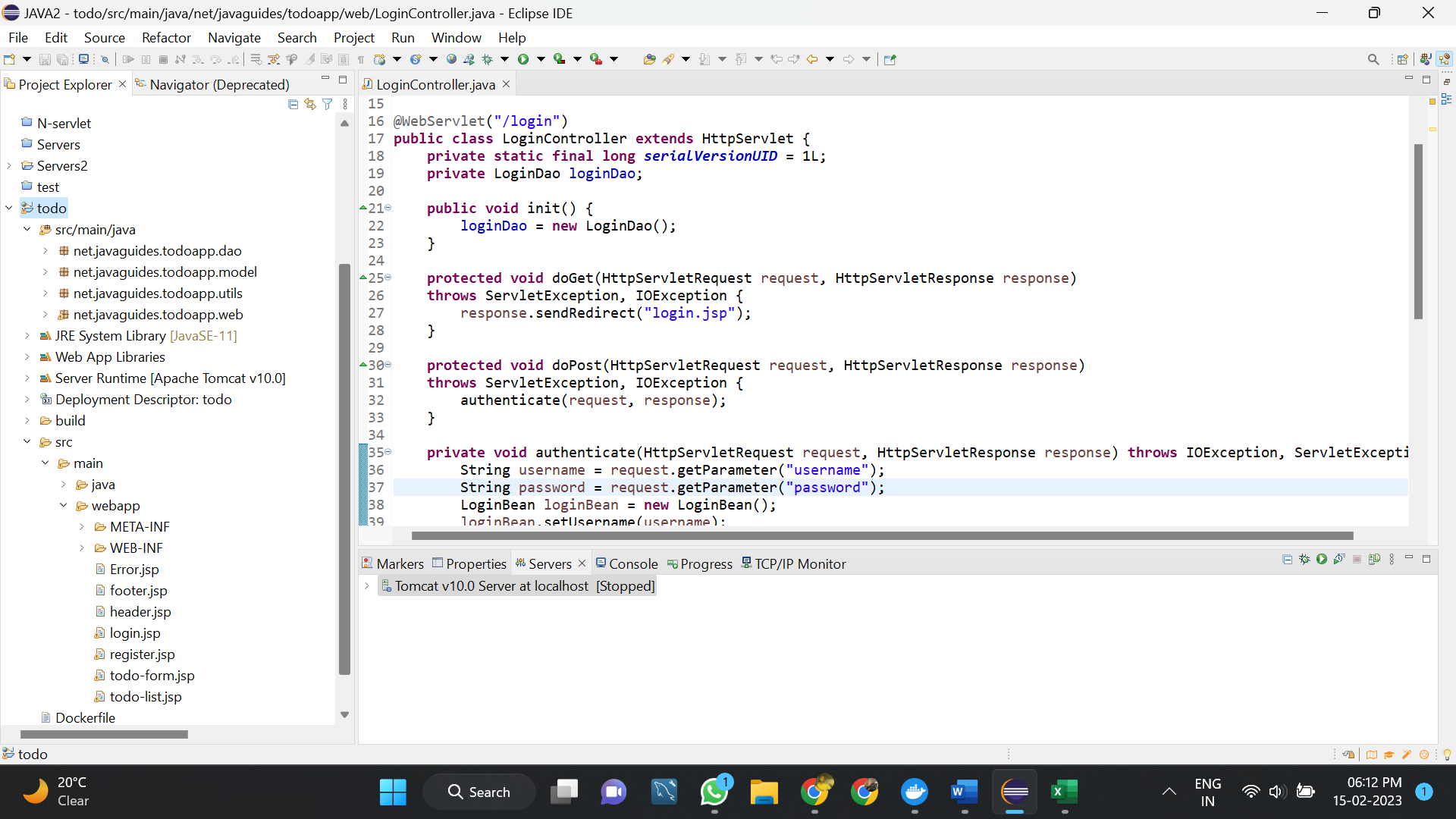
**3.1 IMPLEMENTATION & CODING**

The code for the To Do List in Java project is available on GitHub in a high-quality format. The repository contains all the necessary code files, including the Java classes, JSP pages, and SQL scripts, as well as any required libraries and dependencies. The code is well-documented and follows industry-standard coding practices, making it easy for other developers to understand and contribute to the project. Additionally, the code is regularly maintained and updated to ensure its quality and reliability. Interested developers can access the GitHub repository to view, download, and contribute to the codebase."

It's important to use language that conveys the quality and value of the code on GitHub, as this can encourage other developers to view and contribute to the project. Additionally, providing a link to the GitHub repository can make it easy for interested parties to access and explore the code.

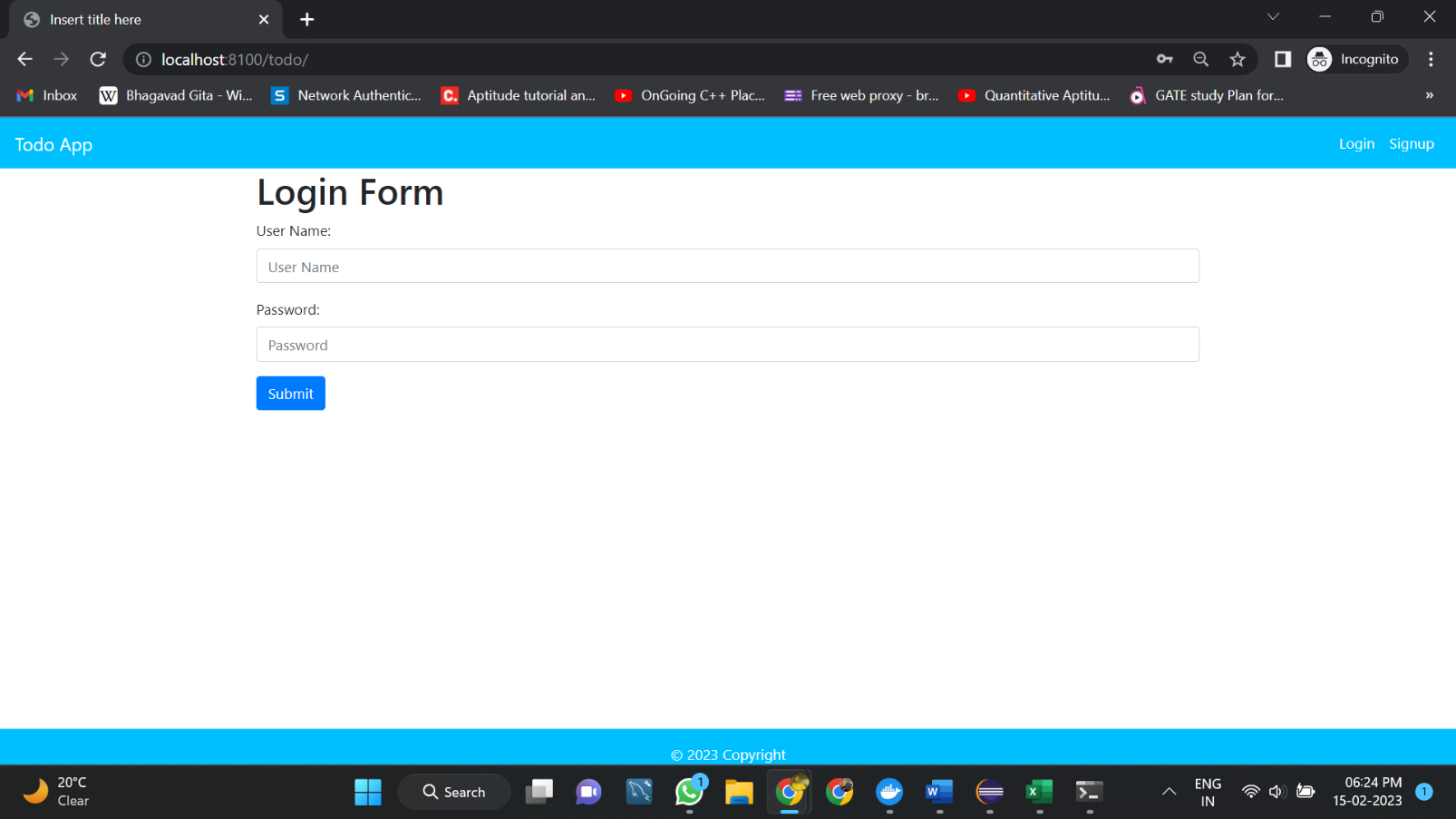
GITHUB LINK -: <https://github.com/Mr-BadalKumar/todo>

To view source code locally in your system: git clone <https://github.com/Mr-BadalKumar/todo.git>

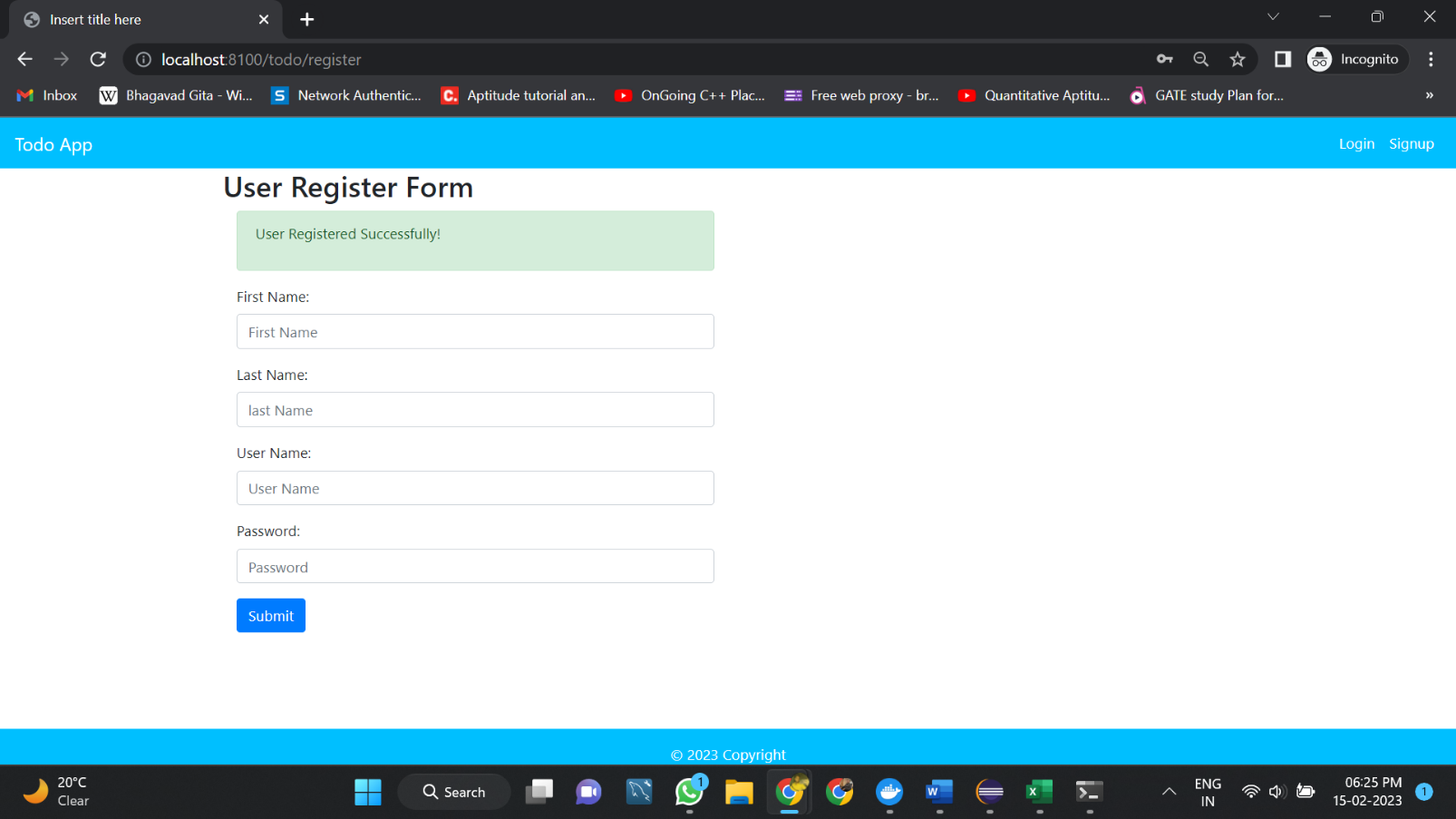


**22**

**LOGINPAGE**

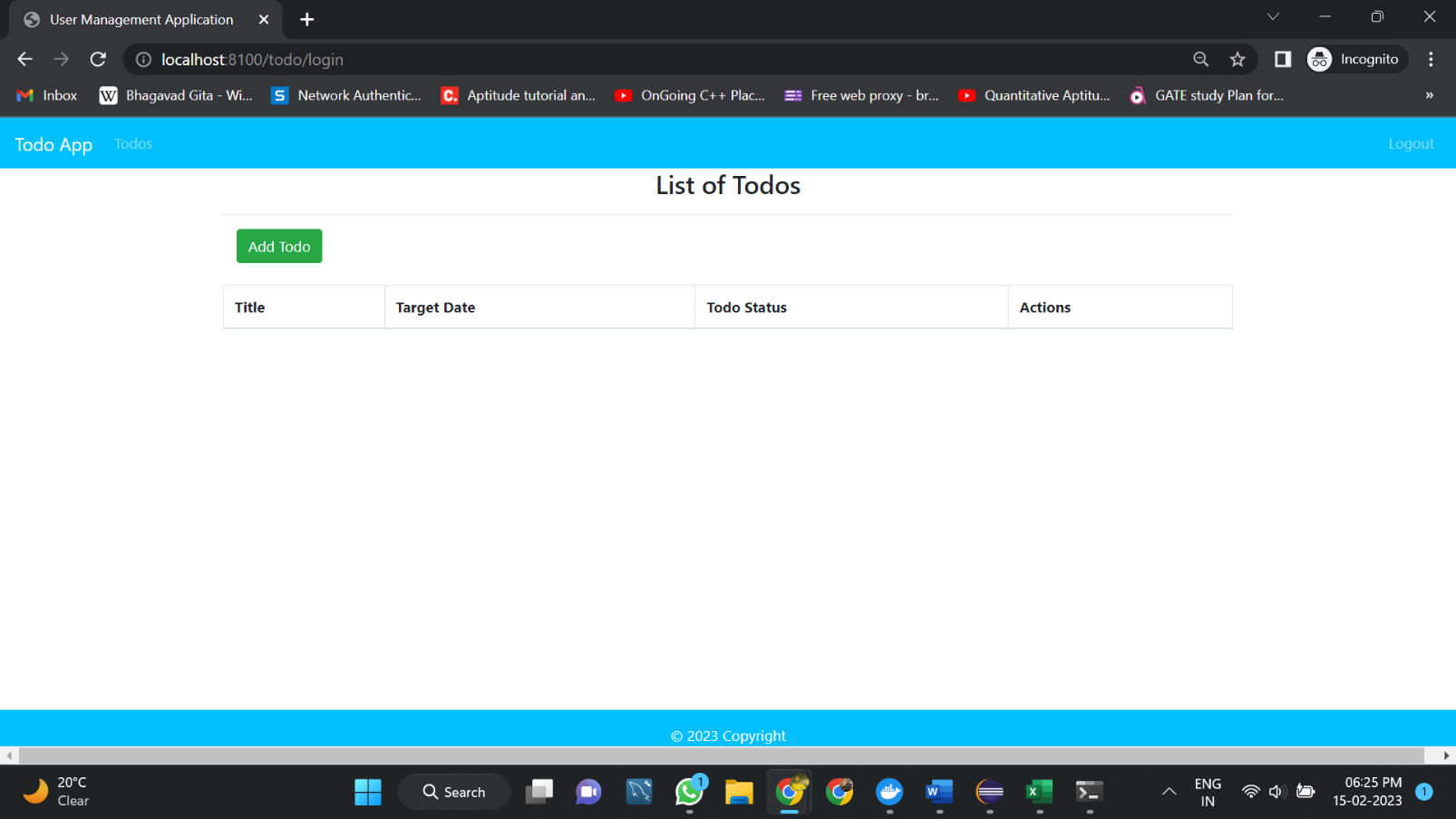


**Registration Page**

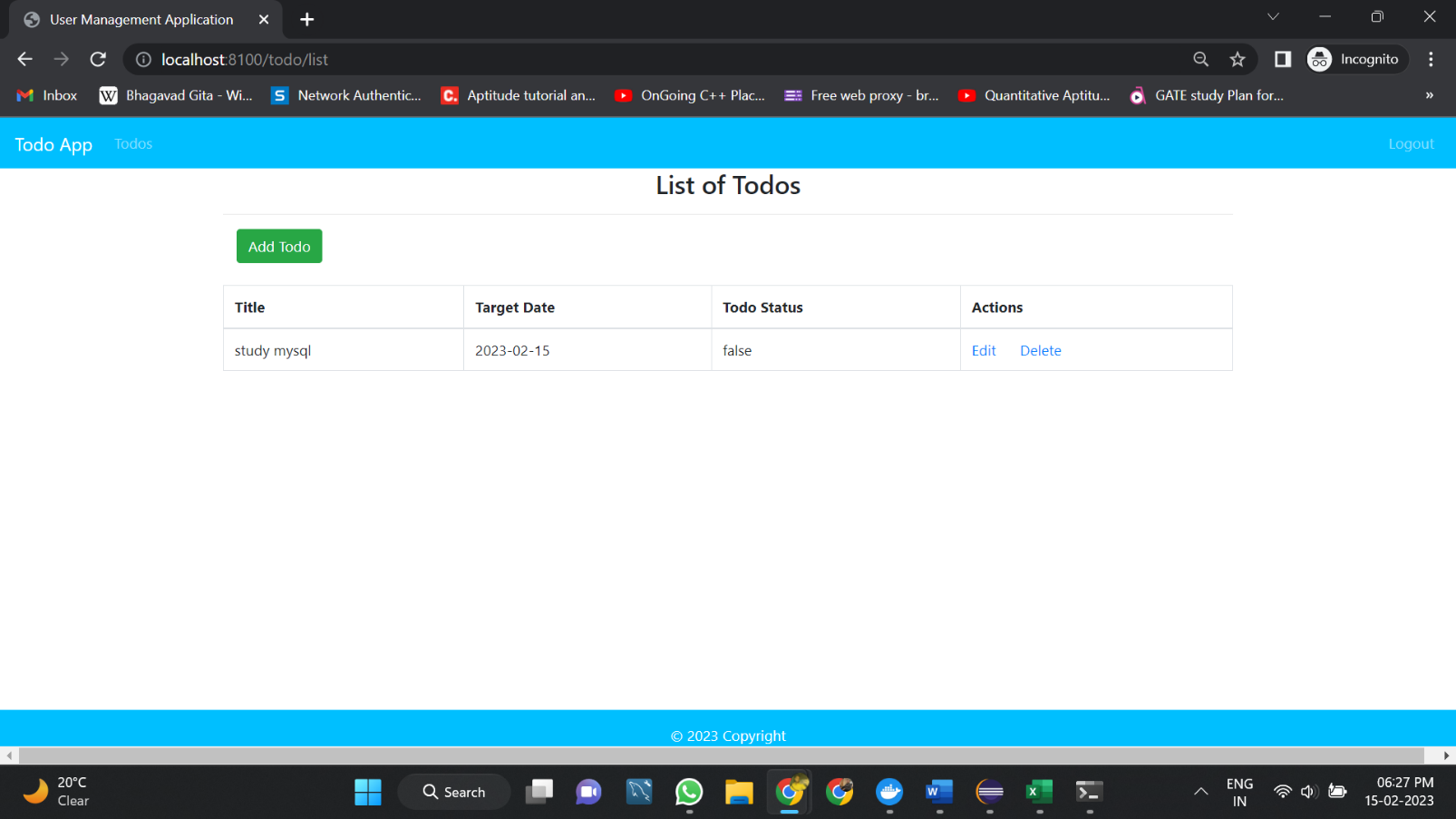


**23**

**TODOLIST**

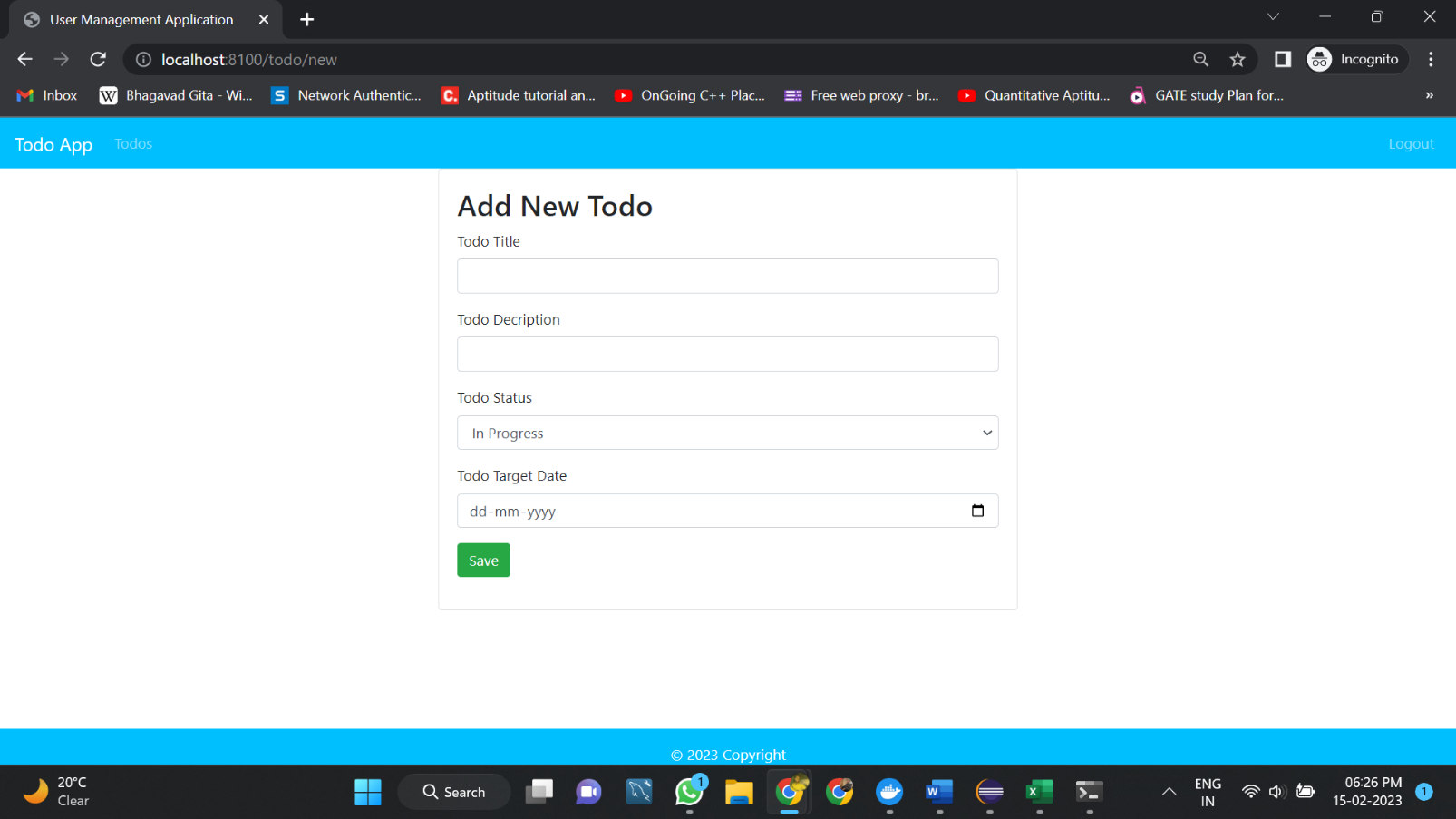


AFTER ADDING TASK HAVING OPTION TO EDIT



**24**

**ADDING TASK PAGE**



**25**

**Chapter 4**

**4.1 TESTING & TESTING RESULTS**

It sounds like you are asking about testing and testing results for a todo list application written in Java using JDBC, Tomcat server, and JSP.

When it comes to testing a web application like a todo list, there are several types of tests you may want to perform, including unit tests, integration tests, and end-to-end tests.

* Unit tests can be used to test individual units of code, such as individual methods or classes, to ensure that they are functioning as intended. For example, you may want to test the methods responsible for adding and deleting items from the todo list.
* Integration tests can be used to test how different components of the application work together. In the case of a web application like a todo list, this might include testing how the JSP pages communicate with the backend Java code, and how the Java code interacts with the database using JDBC.
* End-to-end tests can be used to test the application as a whole, from the user's perspective. This might include testing things like adding items to the todo list, marking items as completed, and deleting items.

To perform these tests, you can use a testing framework like JUnit for unit and integration tests, and a tool like Selenium for end-to-end tests.

Once you have run your tests, you will want to analyze the results to identify any issues or areas for improvement in your application. You may also want to use a tool like JMeter to perform load testing and ensure that your application can handle a large number of users.

Overall, testing is an important part of the software development process, and can help you ensure that your todo list application is reliable and performs well for your users.

**26**

**Chapter 5**

**5.1 RESULTS AND CONCLUSION**

The results and conclusion for a todo list application in Java using JDBC, Tomcat server, and JSP would depend on several factors, such as the goals and requirements of the project, the testing results, and user feedback.

If the application meets the project's requirements and testing results are successful, the conclusion would be that the application is ready for deployment and use. However, if there are issues identified during testing or user feedback, then further development and testing may be necessary before the application can be considered complete.

The testing results can provide valuable insights into the performance, scalability, and usability of the application. For example, load testing can help determine the number of concurrent users the application can handle without experiencing performance issues. At the same time, user testing can help identify any usability issues or areas for improvement in the user interface.

User feedback is also important in evaluating the success of the application. If users find it easy to add and manage tasks and the application performs reliably and efficiently, the application can be considered a success. However, if users have difficulty using the application, or experience technical issues, there may be a need for further development and testing.

Overall, the results and conclusion for a todo list application in Java using JDBC, Tomcat server, and JSP would depend on the quality of the implementation, testing, and user feedback. A successful application would meet the project's requirements, perform efficiently, be scalable, and have a user-friendly interface. On the other hand, if issues arise during testing or user feedback indicates usability issues or technical problems, further development and testing would be necessary before the application could be deployed for use.