A
PROJECT REPORT ON

TODO APPLICATION

By

Sr. No.	NAME	ROLL NO.
1	Sangeeta Singh	32257

GUIDE
MR. ABC

DEPARTMENT OF
ELECTRONICS AND TELECOMMUNICATION ENGINEERING
PUNE INSTITUTE OF COMPUTER TECHNOLOGY
PUNE – 43

A.Y. 2022-23

INDEX

1. PROBLEM STATEMENT:

The problem addressed by the Todo Application is the need for an efficient and user-friendly system to manage personal todo lists. Traditional pen-and-paper methods or generic note-taking applications may lack features specifically tailored for organizing and tracking todos. Therefore, the Todo Application aims to provide a specialized solution that allows users to effectively manage their tasks and enhance productivity.

The key challenges and requirements addressed by the Todo Application are as follows:

1. Task Management: Users need a centralized system to create, organize, and track their tasks. The application should provide features for adding new todos, editing existing ones, and marking tasks as completed.

2. User Authentication: To ensure data privacy and security, the application should implement user authentication. Users should be able to create an account with a unique username and password and securely log in to access their todo list.

3. Responsive Layout: With the increasing use of mobile devices, it is crucial for the application to have a responsive design. The user interface should adapt to different screen sizes and provide a seamless experience across desktops, tablets, and smartphones.

4. Database Integration: The application needs to store todo information and user authentication details in a reliable and scalable database. Integration with MySQL database is required to store and retrieve todos efficiently.

5. Performance and Scalability: The application should be designed and optimized to handle a large number of todos and users. It should efficiently process user requests, perform database operations, and ensure fast response times even under high load.

6. User-Friendly Interface: The user interface should be intuitive, visually appealing, and easy to navigate. Users should be able to quickly add, edit, and delete todos without any confusion or complexity.

7. Error Handling and Validation: The application should handle errors and validate user input to ensure data integrity and prevent unauthorized access. Proper error messages and feedback should be provided to users when they encounter any issues.

The successful implementation of the Todo Application will provide users with an effective and convenient tool to manage their todos, improve productivity, and stay organized. It will address the limitations of traditional methods and provide a user-friendly interface coupled with robust features for seamless task management.

2. OBJECTIVE :

1. Task Management: Develop a system that allows users to create, edit, and delete tasks. Users should be able to assign a title, description, due date, and priority to each task.

2. User Authentication: Implement a secure user authentication system to ensure that only authorized users can access and manage their own todo lists. Users should be able to sign up, log in, and log out of the application.

3. Responsive Layout: Create a responsive user interface that adapts to different screen sizes and devices. The application should provide a seamless user experience on desktops, tablets, and mobile devices.

4. Database Integration: Integrate the application with a MySQL database to store and retrieve todo data. Implement appropriate database schema and queries for efficient data management.

5. Performance and Scalability: Optimize the application for performance and scalability to handle a large number of users and todos. Implement caching mechanisms, query optimizations, and database indexing to improve application responsiveness.

6. User-Friendly Interface: Design an intuitive and user-friendly interface that enables users to easily navigate, add, edit, and delete todos. Provide clear instructions and feedback to users to ensure a smooth user experience.

7. Error Handling and Validation: Implement proper error handling and validation mechanisms to ensure data integrity and prevent unauthorized access. Display informative error messages and validation feedback to users when necessary.

8. Security: Implement appropriate security measures to protect user data and prevent common security vulnerabilities such as SQL injection and cross-site scripting (XSS) attacks.

9. Testing and Debugging: Conduct thorough testing of the application to identify and fix any bugs or issues. Implement logging and debugging mechanisms to facilitate troubleshooting and maintenance.

10. Documentation: Provide comprehensive documentation for the application, including installation instructions, usage guidelines, and code documentation. This will help users and developers understand and utilize the application effectively.

3. INTRODUCTION:

3.1 Background/context

The Todo Application is developed to address the need for an efficient task management system. In today's fast-paced world, individuals and organizations face numerous tasks and deadlines that need to be managed effectively. Traditional methods such as pen and paper or basic to-do lists may not suffice for organizing and tracking tasks, especially when dealing with multiple projects and collaborators.

The application leverages JSP (JavaServer Pages), Servlets, JDBC (Java Database Connectivity), and MySQL database to provide a reliable and scalable solution. JSP and Servlets offer a dynamic web development framework, allowing for server-side processing and generating dynamic web content. JDBC enables seamless communication between the Java application and the MySQL database, facilitating efficient data storage and retrieval.

The application's responsive layout ensures a consistent and user-friendly experience across different devices, including desktops, tablets, and mobile devices. The responsive design adapts to various screen sizes, providing optimal usability and accessibility.

By integrating a secure user authentication system, the application ensures that only authorized users can access and manage their todo lists. User authentication is crucial for data privacy and security, preventing unauthorized access to sensitive information.

The MySQL database serves as the backend storage for todo data. It offers a reliable and scalable solution for storing and retrieving tasks, allowing users to access their todos from anywhere with an internet connection. The database integration enables efficient data management and supports essential features such as adding, editing, and deleting todos.

The Todo Application aims to streamline task management, increase productivity, and improve organization. It provides a centralized platform for users to create, track, and prioritize tasks effectively. With its user-friendly interface, responsive layout, and robust

features, the Todo Application is designed to meet the demands of individuals and organizations seeking an efficient and reliable task management solution.

3.2 Relevance

The relevance of the Todo Application to the field of Electronics and Communication Engineering and related subjects may not be direct or apparent, as the project focuses on task management and web development. However, we can identify some indirect connections and benefits that make the project relevant to the field. These include:

1. Project Management: Electronics and Communication Engineering projects often involve multiple tasks, deadlines, and team coordination. The Todo Application can be utilized by engineering students and professionals to manage their project-related tasks effectively. It enables them to create task lists, set priorities, and track progress, contributing to better project management and organization.

2. Time Management: Time management is crucial in any engineering discipline, including Electronics and Communication Engineering. The Todo Application helps users prioritize their tasks, set deadlines, and allocate time efficiently. By using the application, engineering students can optimize their study schedules, allocate time for practical experiments, and manage their academic and personal commitments more effectively.

3. Documentation and Reporting: Throughout their academic and professional careers, Electronics and Communication Engineering students and professionals need to maintain proper documentation and report their progress. The Todo Application can assist in this regard by providing a platform to create and update task-based documentation. Users can add comments, notes, and attachments to each task, facilitating the recording of project milestones, experiments, and results.

4. Collaboration and Communication: The Todo Application supports team collaboration, which is crucial in many engineering projects. Students can create shared todo lists, assign tasks to team members, and track their progress. This promotes effective communication,

coordination, and accountability within the team, fostering a collaborative working environment.

3.3 Project Details

The project details for the Todo Application are as follows:

1. Project Title: Todo Application
2. Objective: The objective of the project is to develop a web-based application that allows users to manage their tasks and todos effectively. The application will provide features such as adding new todos, editing existing todos, listing todos, and deleting todos. Additionally, it will include user authentication functionality with login and signup features.
3. Technology Stack:
   - Java 8+: The project will be developed using Java programming language, specifically Java 8 or a higher version.
   - JSP and Servlet: JavaServer Pages (JSP) will be used to create the dynamic web pages, and Servlets will handle the server-side processing.
   - JDBC: Java Database Connectivity (JDBC) will be used to interact with the MySQL database.
   - MySQL Database: The application will utilize a MySQL database to store user information and todo data.
   - Eclipse IDE: Eclipse IDE will be used as the development environment for writing and managing the project code.
   - Tomcat Server 8+: The project will be deployed on the Apache Tomcat server, version 8 or a higher version.
4. Features:
   - Add Todo: Users will be able to add new todos by providing a title, description, and due date for each task.
   - Edit Todo: Users can edit the details of existing todos, such as modifying the title, description, or due date.
   - List Todo: The application will display a list of todos, showing the title, description, due date, and completion status.
   - Delete Todo: Users will have the option to delete unwanted todos from the list.
   - Login: The application will provide user authentication functionality, allowing users to log in using their credentials.
   - Signup: New users can create an account by signing up with their details.
   - Responsive Layout: The web application will have a responsive design, ensuring optimal display and usability across different devices and screen sizes.
5. Implementation:
   - The project will be implemented using a layered architecture, separating the presentation layer (JSP), the business logic layer (Servlets), and the data access layer (JDBC).
   - User input will be validated to ensure data integrity and security.
   - Proper exception handling and error reporting mechanisms will be implemented.

6. Testing: The application will undergo comprehensive testing to ensure its functionality, usability, and performance. Test cases will be designed to cover all features and scenarios, including positive and negative cases.
7. Documentation: Documentation will be provided, including a project report that describes the project's purpose, features, implementation details, and testing process. Additionally, documentation for installation, configuration, and usage of the application will be included.
8. Timeline: A timeline will be defined for each phase of the project, including requirements gathering, design, development, testing, and documentation. Milestones and deadlines will be established to track progress and ensure timely completion.

The project details outlined above provide an overview of the Todo Application, including its objectives, technology stack, features, implementation approach, testing process, and documentation.

3.4 Scope**:**

The scope of the Todo Application project is to develop a web-based application that allows users to manage their tasks and todos. The application will provide features such as adding new todos, editing existing todos, listing todos, and deleting todos. It will also include user authentication functionality with login and signup features. The project will be implemented using Java 8+, JSP and Servlets, JDBC, and MySQL database. The application will be deployed on the Tomcat server and will have a responsive layout. The scope includes the development, testing, and documentation of the application.

4.SOURCE CODE:

User Registration Module

1. Create a JavaBean - User.java

```
package net.javaguides.todoapp.model;

import java.io.Serializable;

/**
 * JavaBean class used in jsp action tags.
 * @author Ramesh Fadatare
 */
public class User implements Serializable {
    private static final long serialVersionUID = 1 L;
    private String firstName;
    private String lastName;
    private String username;
    private String password;
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
```

```
}
```

2. Configure JDBC Connection- JDBCUtils.java

```
package net.javaguides.todoapp.utils;

import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.time.LocalDate;

public class JDBCUtils {

    private static String jdbcURL = "jdbc:mysql://localhost:3306/demo";
    private static String jdbcUsername = "root";
    private static String jdbcPassword = "root";

    public static Connection getConnection() {
        Connection connection = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return connection;
    }

    public static void printSQLException(SQLException ex) {
        for (Throwable e: ex) {
            if (e instanceof SQLException) {
                e.printStackTrace(System.err);
                System.err.println("SQLState: " + ((SQLException) e).getSQLState());
                System.err.println("Error Code: " + ((SQLException) e).getErrorCode());
                System.err.println("Message: " + e.getMessage());
                Throwable t = ex.getCause();
                while (t != null) {
                    System.out.println("Cause: " + t);
                    t = t.getCause();
```

```
            }
        }
    }
}

public static Date getSQLDate(LocalDate date) {
    return java.sql.Date.valueOf(date);
}

public static LocalDate getUtilDate(Date sqlDate) {
    return sqlDate.toLocalDate();
}
}
```

Note that we need to provide your MySQL database credentials in the above file.

3. DAO Layer - UserDao.java

Let's create a *UserDao* class which contains JDBC code to store User registered data into the *users* table. Add the following code to an *UserDao* class:

```
package net.javaguides.todoapp.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import net.javaguides.todoapp.model.User;
import net.javaguides.todoapp.utils.JDBCUtils;

public class UserDao {

    public int registerEmployee(User employee) throws ClassNotFoundException {
        String INSERT_USERS_SQL = "INSERT INTO users" +
            "  (first_name, last_name, username, password) VALUES " +
            " (?, ?, ?, ?);";

        int result = 0;
        try (Connection connection = JDBCUtils.getConnection();
            // Step 2:Create a statement using connection object
            PreparedStatement preparedStatement =
connection.prepareStatement(INSERT_USERS_SQL)) {
            preparedStatement.setString(1, employee.getFirstName());
            preparedStatement.setString(2, employee.getLastName());
            preparedStatement.setString(3, employee.getUsername());
            preparedStatement.setString(4, employee.getPassword());
```

```
            System.out.println(preparedStatement);
            // Step 3: Execute the query or update query
            result = preparedStatement.executeUpdate();

        } catch (SQLException e) {
            // process sql exception
            JDBCUtils.printSQLException(e);
        }
        return result;
    }
}
```

5. View Layer - register.jsp

header.jsp

```
<header>
 <nav class="navbar navbar-expand-md navbar-dark"
  style="background-color: tomato">
  <div>
   <a href="https://www.javaguides.net" class="navbar-brand"> Todo App</a>
  </div>

  <ul class="navbar-nav navbar-collapse justify-content-end">
   <li><a href="<%= request.getContextPath() %>/login" class="nav-link">Login</a></li>
   <li><a href="<%= request.getContextPath() %>/register" class="nav-link">Signup</a></li>
  </ul>
 </nav>
</header>
```

footer.jsp

```
<style>
.footer {
    position: fixed;
    bottom: 0;
    width:100%;
    height: 40px;
    background-color: tomato;
}
```

```
</style>

<footer class="footer font-small black">
   <!-- Copyright -->
  <div class="footer-copyright text-center py-3" style="color: white">© 2019 Copyright:
      <a href="https://www.javaguides.net/" > <strong> Java Guides </strong></a>
     </div>
</footer>
<!-- Footer -->
```

register.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
   <!DOCTYPE html>
   <html>

   <head>
      <meta charset="ISO-8859-1">
      <title>Insert title here</title>

      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
   </head>

   </head>

   <body>
      <jsp:include page="../common/header.jsp"></jsp:include>
      <div class="container">

         <h2>User Register Form</h2>
         <div class="col-md-6 col-md-offset-3">
            <div class="alert alert-success center" role="alert">
               <p>${NOTIFICATION}</p>
            </div>

            <form action="<%=request.getContextPath()%>/register" method="post">

               <div class="form-group">
```

```
                <label for="uname">First Name:</label> <input type="text" class="form-control"
id="uname" placeholder="First Name" name="firstName" required>
            </div>

            <div class="form-group">
                <label for="uname">Last Name:</label> <input type="text" class="form-control"
id="uname" placeholder="last Name" name="lastName" required>
            </div>

            <div class="form-group">
                <label for="uname">User Name:</label> <input type="text" class="form-control"
id="username" placeholder="User Name" name="username" required>
            </div>

            <div class="form-group">
                <label for="uname">Password:</label> <input type="password" class="form-
control" id="password" placeholder="Password" name="password" required>
            </div>

            <button type="submit" class="btn btn-primary">Submit</button>

        </form>
      </div>
    </div>
    <jsp:include page="../common/footer.jsp"></jsp:include>
  </body>

  </html>
```

Login Module

1. Create a JavaBean - LoginBean.java

```
package net.javaguides.todoapp.model;

import java.io.Serializable;

public class LoginBean implements Serializable {
  private static final long serialVersionUID = 1 L;
  private String username;
  private String password;

  public String getUsername() {
    return username;
```

```
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

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

2. DAO Layer - LoginDao.java

```
package net.javaguides.todoapp.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import net.javaguides.todoapp.model.LoginBean;
import net.javaguides.todoapp.utils.JDBCUtils;

public class LoginDao {

    public boolean validate(LoginBean loginBean) throws ClassNotFoundException {
        boolean status = false;

        Class.forName("com.mysql.jdbc.Driver");

        try (Connection connection = JDBCUtils.getConnection();
            // Step 2:Create a statement using connection object
            PreparedStatement preparedStatement = connection
            .prepareStatement("select * from users where username = ? and password = ? ")) {
            preparedStatement.setString(1, loginBean.getUsername());
            preparedStatement.setString(2, loginBean.getPassword());

            System.out.println(preparedStatement);
            ResultSet rs = preparedStatement.executeQuery();
            status = rs.next();
```

```
        } catch (SQLException e) {
            // process sql exception
            JDBCUtils.printSQLException(e);
        }
        return status;
    }
}
```

3. Controller Layer - LoginController.java

```
package net.javaguides.todoapp.web;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import net.javaguides.todoapp.dao.LoginDao;
import net.javaguides.todoapp.model.LoginBean;

/**
 * @email Ramesh Fadatare
 */

@WebServlet("/login")
public class LoginController extends HttpServlet {
    private static final long serialVersionUID = 1 L;
    private LoginDao loginDao;

    public void init() {
        loginDao = new LoginDao();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.sendRedirect("login/login.jsp");
    }
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        authenticate(request, response);
    }

    private void authenticate(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        LoginBean loginBean = new LoginBean();
        loginBean.setUsername(username);
        loginBean.setPassword(password);

        try {
            if (loginDao.validate(loginBean)) {
                RequestDispatcher dispatcher = request.getRequestDispatcher("todo/todo-list.jsp");
                dispatcher.forward(request, response);
            } else {
                HttpSession session = request.getSession();
                // session.setAttribute("user", username);
                // response.sendRedirect("login.jsp");
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }

    }
}
```

4. View Layer - login.jsp

Let's design login HTML form with following fields:

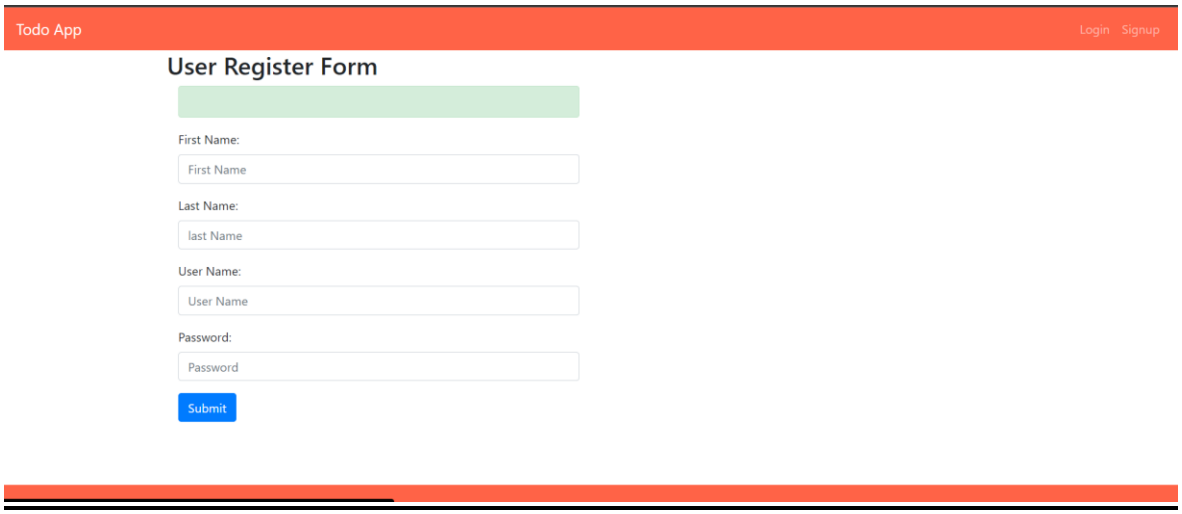
- username
- password

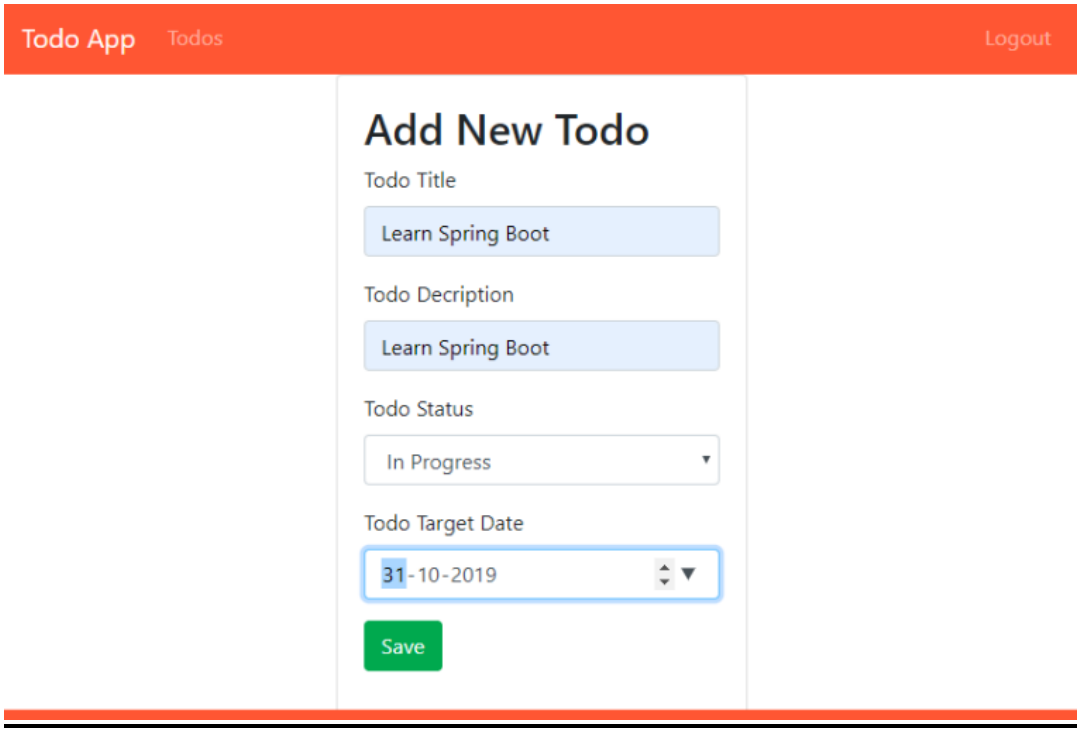
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
    <!DOCTYPE html>
    <html>

    <head>
        <meta charset="ISO-8859-1">
        <title>Insert title here</title>
```

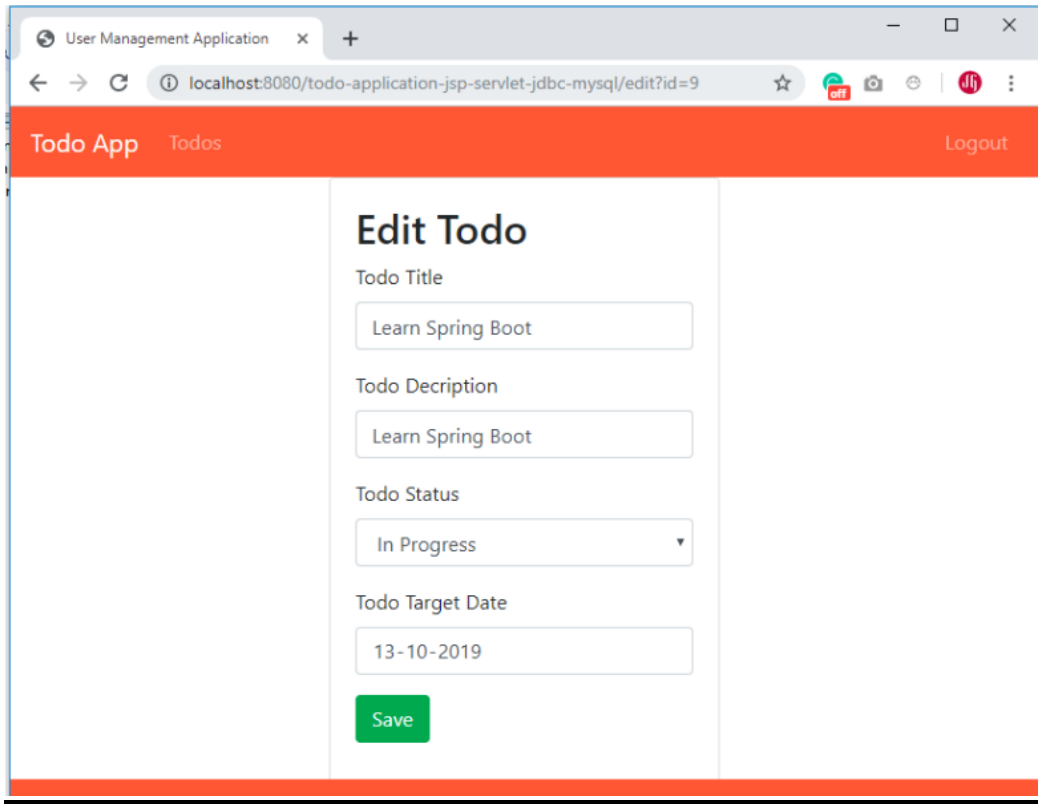
```
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    </head>

    <body>

        <jsp:include page="../common/header.jsp"></jsp:include>
        <div class="container col-md-8 col-md-offset-3" style="overflow: auto">
            <h1>Login Form</h1>
            <form action="<%=request.getContextPath()%>/login" method="post">

                <div class="form-group">
                    <label for="uname">User Name:</label> <input type="text" class="form-control"
id="username" placeholder="User Name" name="username" required>
                </div>

                <div class="form-group">
                    <label for="uname">Password:</label> <input type="password" class="form-
control" id="password" placeholder="Password" name="password" required>
                </div>


                <button type="submit" class="btn btn-primary">Submit</button>
            </form>
        </div>
        <jsp:include page="../common/footer.jsp"></jsp:include>
    </body>

    </html>
```

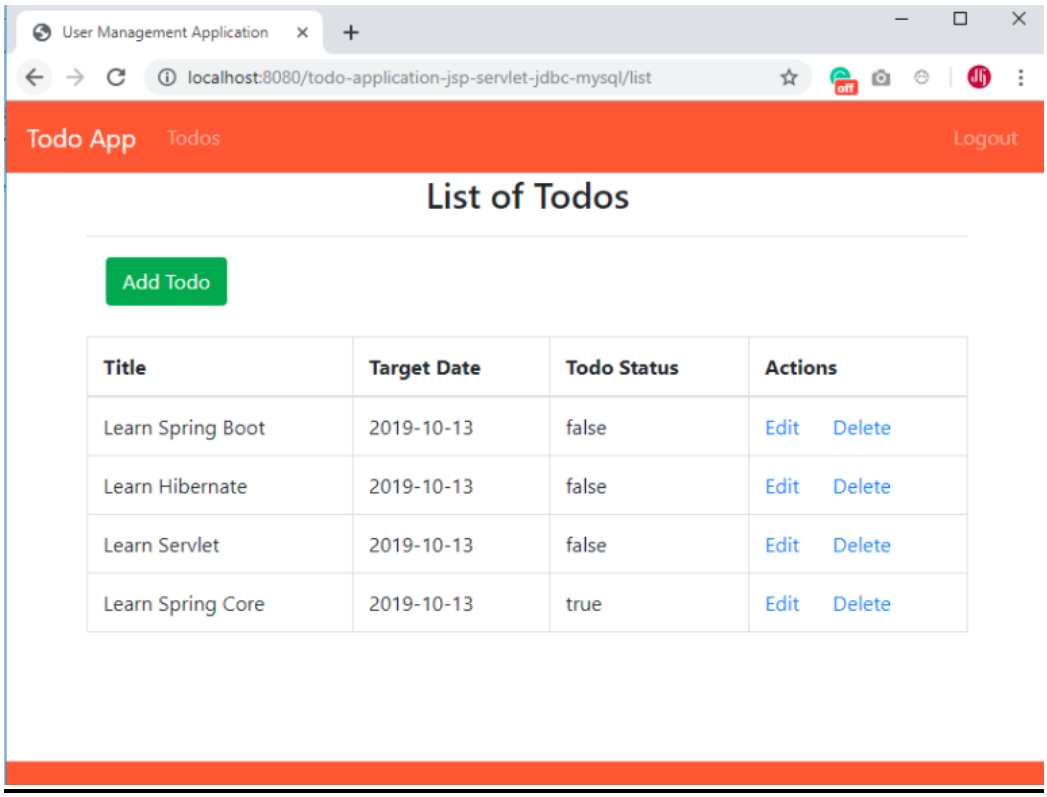
5.RESULT:


Todo App
Login
Signup
Login Form
User Name:
tester
Password:
Password
Submit


Todo App
Login
Signup
User Register Form
First Name:
First Name
Last Name:
last Name
User Name:
User Name
Password:
Password
Submit


Todo App
Todos
Logout
Add New Todo
Todo Title
Learn Spring Boot
Todo Decription
Learn Spring Boot
Todo Status
In Progress
Todo Target Date
31-10-2019
Save


User Management Application
localhost:8080/todo-application-jsp-servlet-jdbc-mysql/edit?id=9
Todo App
Todos
Logout
Edit Todo
Todo Title
Learn Spring Boot
Todo Decription
Learn Spring Boot
Todo Status
In Progress
Todo Target Date
13-10-2019
Save



6. CONCLUSION:

In conclusion, the Todo Application project serves as a practical and relevant example of using Java, JSP, Servlets, JDBC, and MySQL database to develop a web-based task management system. The project covers essential features such as adding, editing, listing, and deleting todos, as well as user authentication functionalities. It aligns with the field of Electronics and Communication Engineering by demonstrating the utilization of programming languages, databases, and web technologies commonly encountered in software development.

By successfully completing this project, learners will gain hands-on experience in developing a web application, understanding the interactions between the frontend and backend components, and managing data using a database system. Additionally, they will become familiar with concepts such as session management, form handling, database connectivity, and user authentication.

Overall, the Todo Application project provides an opportunity for students and enthusiasts of Electronics and Communication Engineering to apply their theoretical knowledge and enhance their practical skills in the domain of web development. It encourages them to think critically, solve problems, and develop a functional application that can be further extended and customized based on specific requirements

7.APPLICATIONS:

The Todo Application project has various applications in real-world scenarios. Some of the potential applications include:

1. Task Management Systems: The project can be utilized as a foundation for developing comprehensive task management systems used in organizations and businesses. It provides the basic functionalities of creating, editing, and deleting tasks, making it suitable for building more advanced task management applications.

2. Project Management Tools: The project can be extended to incorporate project management features, such as assigning tasks to team members, setting deadlines, tracking progress, and generating reports. This would make it useful for managing projects in various industries.

3. Personal Task Organizers: Individuals can use the application to manage their personal tasks and to-do lists. They can add, edit, and prioritize tasks, ensuring efficient time management and organization.

4. Collaborative Task Management: By implementing user management and access control features, the project can be expanded into a collaborative task management platform. Multiple users can collaborate on tasks, assign responsibilities, and monitor progress, making it ideal for team-based projects.

5. Educational Platforms: The project can be adapted to serve as an educational platform for teaching programming concepts, web development, and database management. It can provide a hands-on learning experience for students, allowing them to understand the practical aspects of building web applications.

6. Workflow Automation: The application can be integrated into existing workflow systems to automate task assignment and tracking. This would streamline processes and improve efficiency in industries such as project management, customer support, and content publishing.

7. Personal Productivity Tools: Individuals can customize the project to suit their specific productivity needs. They can add features like reminders, task categorization, priority levels, and notifications to enhance their personal productivity and time management.

Overall, the Todo Application project has versatile applications across various industries and personal use cases. Its flexibility allows for customization and adaptation to specific requirements, making it a valuable tool for task and project management.

8. FUTURE SCOPE:

The Todo Application project has several future scope opportunities for further enhancements and expansion. Some potential areas of future development include:

1. Mobile Application: The project can be extended to develop a mobile version of the application, allowing users to manage their tasks on-the-go. This would involve designing a responsive and user-friendly mobile interface and implementing features specific to mobile platforms.

2. Notifications and Reminders: Implementing a notification system would enable users to receive reminders for upcoming tasks, deadlines, or important updates. This can be achieved through email notifications, push notifications on mobile devices, or SMS alerts.

3. Task Categorization and Filtering: Enhancing the application with the ability to categorize tasks and apply filters would provide users with better organization and easy retrieval of specific tasks. Users can categorize tasks based on priority, due date, project, or any custom criteria.

4. Collaboration and Sharing: Introducing collaboration features would enable users to share tasks, assign tasks to team members, and track progress collectively. This would facilitate teamwork and improve coordination in projects involving multiple individuals.

5. Data Analytics and Reporting: Implementing data analytics capabilities would allow users to generate reports and gain insights from their task data. Users can analyze their task completion rates, identify patterns, and make informed decisions based on the collected data.

6. Integration with External Tools and Services: Integrating the application with external tools and services, such as calendar applications, project management software, or cloud storage platforms, would enhance its functionality and interoperability.

7. User Experience Enhancements: Continuously improving the user interface (UI) and user experience (UX) of the application can contribute to its success. This involves incorporating intuitive design, responsiveness, accessibility, and smooth navigation.

8. Integration with Chatbots or Virtual Assistants: Integrating the application with chatbots or virtual assistants would provide users with an interactive and conversational experience. Users can interact with the application using voice commands or natural language, simplifying task management.

9. Gamification and Rewards: Adding gamification elements, such as badges, achievements, or rewards, can make task management more engaging and motivate users to complete their tasks in a timely manner.

10. Data Synchronization and Cloud Storage: Implementing data synchronization across devices and integrating with cloud storage services would ensure that users can access their tasks from multiple devices and have their data securely backed up.

These future scope opportunities can enhance the functionality, usability, and user engagement of the Todo Application project, making it more versatile and appealing to a wider range of users.

9. COPY RIGHT AFFIRMATION:

We undersigned pledge and represent that the source code printed in this mini project report does not violate any proprietary or personal rights of others (including, without limitation, any copyrights or privacy rights); that the Work is factually accurate and contains no matter libellous or otherwise unlawful; that we have substantially participated in the creation of the Work and that it represents our original work sufficient for us to claim authorship.

Name of students **Sign**

1. **Sangeeta Singh**