DRONACHARYA
Group of Institutions

B-27, Knowledge Park – III, Greater Noida Uttar Pradesh - 201308
Approved by: All India Council for Technical Education (AICTE), New Delhi
Affiliated to: Dr. A. P. J. Abdul Kalam Technical University (AKTU), Lucknow

DEPARTMENT OF INFORMATION TECHNOLOGY

Web Technology Lab Manual

SUBJECT CODE: BCS-552

B.Tech., Semester -V

Session: 2024-25, ODD Semester

Table of Contents

DRONACHARYA GROUP OF INSTITUTIONS GREATER NOIDA

VISION

- Instilling core human values and facilitating competence to address global challenges by providing Quality Technical Education.

MISSION

- M1 - Enhancing technical expertise through innovative research and education, fostering creativity and excellence in problem-solving.
- M2 - Cultivating a culture of ethical innovation and user-focused design, ensuring technological progress enhances the well-being of society.
- M3 - Equipping individuals with the technical skills and ethical values to lead and innovate responsibly in an ever-evolving digital land.

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION

To provide students with theoretical understanding and technical proficiency in Information Technology, instill with moral and ethical values, to excel in academic, industry, and research settings.

MISSION

M1: To instill in students a strong foundation of both the theory and practical application of IT skills, combined with the innovation and research approaches to keep pace with emerging technologies.

M2: To empower graduates to become global leaders specializing in field of Information Technology.

M3: To impart to students the social, ethical, and moral values necessary for them to make substantial contributions to society.

Program Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO 9: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply theseto one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Programme Educational Objectives (PEOs)

PEO1: Apply the knowledge of mathematics, science and engineering fundamentals to identify and solve IT and engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

Program Specific Outcomes (PSOs)

PSO1: Ability to think logically and apply programming knowledge and practices in analyzing real world problems and provide solutions to meet the needs of society.

PSO2: Enhance the competence of technocrats to provide professional engineering solutions as per the industrial and societal needs.

PSO3: To cultivate and produce highly motivated engineers committed to lifelong learning, pursuing research, higher education, and embracing competitive challenges to excel in the future.

CO – PO Mapping

BCS-552. 1	Practice web designing in technologies like CSS and Bootstrap .
BCS-552. 2	Analyze the data handling layers and security to data flow among different pages.
BCS-552. 3	Specification of servers and communication protocols.
BCS-552. 4	Demonstrate various communication strategy and layers of networking.
BCS-552. 5	Elaborating the connection with various database and encryption methodology

CO-PSO Mapping:

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO 1	3	2	3	2	2	-	-	-	-	1	1	3
CO 2	3	3	2	-	2	-	-	-	-	1	1	2
CO 3	3	2	3	3	2	-	-	-	-	1	1	3
CO 4	2	3	3	3	2	-	-	-	-	1	1	2
CO 5	1	3	3	-	2	-	-	-	-	1	1	3

PSO1	**PSO2**	**PSO3**
1	2	1
1	3	1
1	3	1
2	2	1
2	2	1

Course Overview

The students will be able to develop and design web applications. Students shall be able to handle server and hosting strategy of various organizations. Students will be able to implement the enhanced security structure in communication with server.

Course Objectives

- To introduce the fundamental concept of web technology and security
- To emphasize the importance of client server model with various layers of security
- To develop effective skills in the implementation of encryption in data

BCS552- Web Technology Lab

List of Experiments (Indicative & not limited to)

1. Write HTML program for designing your institute website. Display departmental information of your institute on the website.
2. Write HTML program to design an entry form for student details/employee information/faculty details.
3. Develop a responsive website using CSS and HTML. Website may be for tutorial/blogs/commercial website.
4. Write programs using HTML and Java Script for validation of input data.
5. Write a program in XML for creation of DTD, which specifies set of rules. Create a style sheet in CSS/ XSL & display the document in internet explorer.
6. Create a Java Bean for Employee information (EmpID, Name, Salary, Designation and Department).
7. Build a command-line utility using Node.js that performs a specific task, such as converting text to uppercase, calculating the factorial of a number, or generating random passwords.
8. Develop a script that uses MongoDB's aggregation framework to perform operations like grouping, filtering, and sorting. For instance, aggregate user data to find the average age of users in different cities.
9. Assume four users user1, user2, user3 and user4 having the passwords pwd1, pwd2, pwd3 and pwd4 respectively. Write a servlet for doing the following: 1. Create a Cookie and add these four user id's and passwords to this Cookie. 2. Read the user id and passwords entered in the Login form and authenticate with the values available in the cookies.
10. Create a table which should contain at least the following fields: name, password, email-id, phone number Write Servlet/JSP to connect to that database and extract data from the tables and display them. Insert the details of the users who register with the web site, whenever a new user clicks the submit button in the registration page.
11. Write a JSP which insert the details of the 3 or 4 users who register with the web site by using registration form. Authenticate the user when he submits the login form using the user name and password from the database.
12. Design and implement a simple shopping cart example with session tracking API.

LIST OF PROGRAMS

S. No	Aim of the	COs
1.	Write HTML program for designing your institute website. Display departmental information of your institute on the website.	CO1
2.	Write HTML program to design an entry form for student details/employee information/faculty details.	CO4

3.	Develop a responsive website using CSS and HTML. Website may be for tutorial/blogs/commercial website.	CO1
4.	Write programs using HTML and Java Script for validation of input data.	CO2
5.	Write a program in XML for creation of DTD, which specifies set of rules. Create a style sheet in CSS/ XSL & display the document in internet explorer.	CO3
6.	Create a Java Bean for Employee information (EmpID, Name, Salary, Designation and Department).	CO3
7.	Build a command-line utility using Node.js that performs a specific task, such as converting text to uppercase, calculating the factorial of a number, or generating random passwords.	CO4
8.	Develop a script that uses MongoDB's aggregation framework to perform operations like grouping, filtering, and sorting. For instance, aggregate user data to find the average age of users in different cities.	CO5
9.	Assume four users user1, user2, user3 and user4 having the passwords pwd1, pwd2, pwd3 and pwd4 respectively. Write a servlet for doing the following: 1. Create a Cookie and add these four user id's and passwords to this Cookie. 2. Read the user id and passwords entered in the Login form and authenticate with the values available in the cookies.	CO2
10	Create a table which should contain at least the following fields: name, password, email-id, phone number Write Servlet/JSP to connect to that database and extract data from the tables and display them. Insert the details of the users who register with the web site, whenever a new user clicks the submit button in the registration page.	CO2
11	Write a JSP which insert the details of the 3 or 4 users who register with the web site by using registration form. Authenticate the user when he submits the login form using the user name and password from the database.	CO4
12	Design and implement a simple shopping cart example with session tracking API.	CO5

Experiment plan

Assignment: 1
Aim:- Write HTML program for designing your institute website. Display departmental information of your institute on the website.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Dronacharya Institute</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
```

```
            padding: 0;
            background-color: #f2f2f2;
            color: #333;
        }
        header {
            background-color: #004080;
            color: #ffffff;
            padding: 20px;
            text-align: center;
        }
        nav {
            background-color: #333;
            overflow: hidden;
        }
        nav a {
            float: left;
            display: block;
            color: #ffffff;
            text-align: center;
            padding: 14px 20px;
            text-decoration: none;
        }
        nav a:hover {
            background-color: #575757;
        }
        .content {
            padding: 20px;
            text-align: center;
        }
        .department {
            border: 1px solid #cccccc;
            padding: 15px;
            margin: 10px;
            background-color: #ffffff;
            border-radius: 8px;
            box-shadow: 2px 2px 8px rgba(0, 0, 0, 0.1);
        }
        .footer {
            background-color: #004080;
            color: #ffffff;
            text-align: center;
            padding: 10px;
            position: fixed;
            bottom: 0;
            width: 100%;
        }
    </style>
</head>
<body>
```

```
<header>
   <h1>Welcome to Dronacharya Institute</h1>
   <p>Your Gateway to Quality Education</p>
</header>

<nav>
   <a href="#home">Home</a>
   <a href="#about">About Us</a>
   <a href="#departments">Departments</a>
   <a href="#contact">Contact Us</a>
</nav>

<div class="content" id="departments">
   <h2>Our Departments</h2>
   <p>Explore the various departments at Dronacharya Institute dedicated to fostering
knowledge and innovation.</p>

   <div class="department">
      <h3>Computer Science and Engineering</h3>
      <p>The CSE department focuses on cutting-edge technologies in software development,
artificial intelligence, and machine learning.</p>
   </div>

   <div class="department">
      <h3>Electronics and Communication Engineering</h3>
      <p>ECE department provides in-depth knowledge of communication systems,
electronics, and microprocessors, preparing students for the electronics industry.</p>
   </div>

   <div class="department">
      <h3>Mechanical Engineering</h3>
      <p>The ME department equips students with knowledge in thermodynamics, materials
science, and robotics, with practical experience in workshops.</p>
   </div>

   <div class="department">
      <h3>Civil Engineering</h3>
      <p>The Civil Engineering department specializes in structural analysis, construction
technology, and environmental engineering.</p>
   </div>

   <div class="department">
      <h3>Electrical Engineering</h3>
      <p>EE department covers topics like power systems, circuit analysis, and electrical
machines, offering hands-on experience in the labs.</p>
   </div>
</div>

<div class="footer">
   <p>&copy; 2023 Dronacharya Institute. All Rights Reserved.</p>
```

```
</div>

</body>
</html>
```

Assignment 2 :
Aim : Write HTML program to design an entry form for student details/employee information/faculty details.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Entry Form</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         display: flex;
         justify-content: center;
         align-items: center;
         height: 100vh;
```

```
            margin: 0;
            background-color: #f3f3f3;
        }
        .container {
            width: 400px;
            padding: 20px;
            background-color: white;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            border-radius: 5px;
        }
        h2 {
            text-align: center;
            color: #333;
        }
        .form-group {
            margin-bottom: 15px;
        }
        label {
            display: block;
            margin-bottom: 5px;
            color: #555;
        }
        input[type="text"],
        input[type="email"],
        input[type="tel"],
        select {
            width: 100%;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        input[type="submit"] {
            width: 100%;
            padding: 10px;
            border: none;
            background-color: #4CAF50;
            color: white;
            font-size: 16px;
            border-radius: 4px;
            cursor: pointer;
        }
        input[type="submit"]:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>

<div class="container">
    <h2>Entry Form</h2>
```

```
    <form action="#" method="POST">
        <!-- Name -->
        <div class="form-group">
            <label for="name">Full Name:</label>
            <input type="text" id="name" name="name" required>
        </div>

        <!-- ID Number -->
        <div class="form-group">
            <label for="id">ID Number:</label>
            <input type="text" id="id" name="id" required>
        </div>

        <!-- Email -->
        <div class="form-group">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required>
        </div>

        <!-- Contact Number -->
        <div class="form-group">
            <label for="phone">Contact Number:</label>
            <input type="tel" id="phone" name="phone" required>
        </div>

        <!-- Role Selection -->
        <div class="form-group">
            <label for="role">Role:</label>
            <select id="role" name="role" required>
                <option value="">Select Role</option>
                <option value="student">Student</option>
                <option value="employee">Employee</option>
                <option value="faculty">Faculty</option>
            </select>
        </div>

        <!-- Department -->
        <div class="form-group">
            <label for="department">Department:</label>
            <input type="text" id="department" name="department">
        </div>

        <!-- Submit Button -->
        <input type="submit" value="Submit">
    </form>
</div>

</body>
</html>
```

Assignment: 3
Aim:- Develop a responsive website using CSS and HTML. Website may be for tutorial/blogs/commercial website.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Responsive Website</title>
   <style>
      /* Basic Reset */
      * {
         margin: 0;
         padding: 0;
         box-sizing: border-box;
      }
```

```
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    color: #333;
    background-color: #f9f9f9;
}

/* Header */
header {
    background-color: #4CAF50;
    color: #fff;
    padding: 20px 0;
    text-align: center;
}

header h1 {
    font-size: 2.5em;
}

/* Navigation */
nav {
    display: flex;
    justify-content: center;
    background: #333;
}

nav ul {
    list-style: none;
    display: flex;
    padding: 10px;
}

nav ul li {
    margin: 0 15px;
}

nav ul li a {
    color: #fff;
    text-decoration: none;
    font-weight: bold;
}

nav ul li a:hover {
    color: #4CAF50;
}

/* Main Content */
.container {
    max-width: 1200px;
```

```
    margin: 20px auto;
    padding: 20px;
}

.content, .sidebar {
    padding: 20px;
    background-color: #fff;
    margin-bottom: 20px;
}

.content {
    flex: 2;
}

.sidebar {
    flex: 1;
}

.row {
    display: flex;
    flex-wrap: wrap;
}

/* Footer */
footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px 0;
}

footer p {
    margin: 0;
}

/* Responsive Design */
@media (max-width: 768px) {
    nav ul {
        flex-direction: column;
        text-align: center;
    }

    .row {
        flex-direction: column;
    }

    .content, .sidebar {
        margin: 0;
    }
}
```

```
    </style>
</head>
<body>

<!-- Header Section -->
<header>
    <h1>Welcome to Our Website</h1>
    <p>Your go-to resource for tutorials, blogs, and more!</p>
</header>

<!-- Navigation -->
<nav>
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Tutorials</a></li>
        <li><a href="#">Blogs</a></li>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Contact</a></li>
    </ul>
</nav>

<!-- Main Content Section -->
<div class="container">
    <div class="row">
        <!-- Main Content -->
        <div class="content">
            <h2>Main Content Area</h2>
            <p>This section is ideal for blog posts, tutorials, or articles. You can update this area with new content that reflects your niche, whether that's technical tutorials, lifestyle blogs, or commercial information.</p>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus lacinia odio vitae vestibulum vestibulum. Cras venenatis euismod malesuada.</p>
        </div>

        <!-- Sidebar -->
        <aside class="sidebar">
            <h3>About Us</h3>
            <p>We provide quality tutorials and resources to help you learn and grow. Whether you are here to read a blog or follow a tutorial, we're here to guide you through.</p>
            <h3>Recent Posts</h3>
            <ul>
                <li><a href="#">How to Learn HTML</a></li>
                <li><a href="#">Getting Started with CSS</a></li>
                <li><a href="#">Top 10 Coding Tips</a></li>
            </ul>
        </aside>
    </div>
</div>

<!-- Footer Section -->
```

```
<footer>
    <p>&copy; 2024 Your Website Name. All rights reserved.</p>
</footer>

</body>
</html>
```

Assignment: 4
Aim:- Write programs using HTML and Java Script for validation of input data.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form Validation</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
```

```
            margin: 0;
            background-color: #f3f3f3;
        }
        .container {
            width: 400px;
            padding: 20px;
            background-color: white;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            border-radius: 5px;
        }
        h2 {
            text-align: center;
            color: #333;
        }
        .form-group {
            margin-bottom: 15px;
        }
        label {
            display: block;
            margin-bottom: 5px;
            color: #555;
        }
        input[type="text"],
        input[type="email"],
        input[type="tel"],
        input[type="password"] {
            width: 100%;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        input[type="submit"] {
            width: 100%;
            padding: 10px;
            border: none;
            background-color: #4CAF50;
            color: white;
            font-size: 16px;
            border-radius: 4px;
            cursor: pointer;
        }
        input[type="submit"]:hover {
            background-color: #45a049;
        }
        .error {
            color: red;
            font-size: 14px;
        }
    </style>
</head>
```

```
<body>

<div class="container">
  <h2>Registration Form</h2>
  <form name="registrationForm" onsubmit="return validateForm()">
    <!-- Name Field -->
    <div class="form-group">
      <label for="name">Full Name:</label>
      <input type="text" id="name" name="name">
      <div id="nameError" class="error"></div>
    </div>

    <!-- Email Field -->
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" id="email" name="email">
      <div id="emailError" class="error"></div>
    </div>

    <!-- Phone Number Field -->
    <div class="form-group">
      <label for="phone">Phone Number:</label>
      <input type="tel" id="phone" name="phone">
      <div id="phoneError" class="error"></div>
    </div>

    <!-- Password Field -->
    <div class="form-group">
      <label for="password">Password:</label>
      <input type="password" id="password" name="password">
      <div id="passwordError" class="error"></div>
    </div>

    <!-- Submit Button -->
    <input type="submit" value="Submit">
  </form>
</div>

<script>
  function validateForm() {
    // Clear previous error messages
    document.getElementById("nameError").innerText = "";
    document.getElementById("emailError").innerText = "";
    document.getElementById("phoneError").innerText = "";
    document.getElementById("passwordError").innerText = "";

    // Retrieve form values
    const name = document.forms["registrationForm"]["name"].value;
    const email = document.forms["registrationForm"]["email"].value;
    const phone = document.forms["registrationForm"]["phone"].value;
```

```
        const password = document.forms["registrationForm"]["password"].value;

        let isValid = true;

        // Validate Name
        if (name === "") {
            document.getElementById("nameError").innerText = "Name is required.";
            isValid = false;
        } else if (name.length < 3) {
            document.getElementById("nameError").innerText = "Name must be at least 3
characters long.";
            isValid = false;
        }

        // Validate Email
        const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (email === "") {
            document.getElementById("emailError").innerText = "Email is required.";
            isValid = false;
        } else if (!emailPattern.test(email)) {
            document.getElementById("emailError").innerText = "Enter a valid email address.";
            isValid = false;
        }

        // Validate Phone Number
        const phonePattern = /^[0-9]{10}$/;
        if (phone === "") {
            document.getElementById("phoneError").innerText = "Phone number is required.";
            isValid = false;
        } else if (!phonePattern.test(phone)) {
            document.getElementById("phoneError").innerText = "Enter a valid 10-digit phone
number.";
            isValid = false;
        }

        // Validate Password
        if (password === "") {
            document.getElementById("passwordError").innerText = "Password is required.";
            isValid = false;
        } else if (password.length < 6) {
            document.getElementById("passwordError").innerText = "Password must be at least
6 characters long.";
            isValid = false;
        }

        return isValid; // Prevent form submission if validation fails
    }
</script>

</body>
```

</html>:
Assignment: 5
Aim:- Write a program in XML for creation of DTD, which specifies set of rules. Create a style sheet in CSS/ XSL & display the document in internet explorer.
Books.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "books.dtd">
<?xml-stylesheet type="text/css" href="styles.css"?>

<books>
  <book>
    <title>Learning XML</title>
    <author>John Doe</author>
    <publisher>OpenAI Publishing</publisher>
    <price>29.99</price>
  </book>
  <book>
    <title>Advanced HTML</title>
    <author>Jane Smith</author>
    <publisher>Tech Books</publisher>
    <price>39.95</price>
  </book>
</books>
```

Books.dtd

```
<!ELEMENT books (book+)>
<!ELEMENT book (title, author, publisher, price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

Style.css

```
/* Styling for the entire books list */
books {
  display: block;
  font-family: Arial, sans-serif;
  margin: 20px;
  padding: 20px;
  border: 1px solid #ddd;
  background-color: #f9f9f9;
}

book {
  display: block;
  margin-bottom: 15px;
  padding: 10px;
  border-bottom: 1px solid #ccc;
}

title {
```

```
    font-size: 20px;
    font-weight: bold;
    color: #333;
}

author, publisher, price {
    display: block;
    margin-top: 5px;
    font-size: 14px;
    color: #555;
}
```

Assignment 6:
Aim:- Create a Java Bean for Employee information (EmpID, Name, Salary, Designation and Department):

```
import java.io.Serializable;

public class Employee implements Serializable {
    // Private fields
    private int empID;
    private String name;
    private double salary;
    private String designation;
    private String department;

    // No-argument constructor
    public Employee() {
```

```
}

// Getters and Setters

public int getEmpID() {
    return empID;
}

public void setEmpID(int empID) {
    this.empID = empID;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}

public String getDesignation() {
    return designation;
}

public void setDesignation(String designation) {
    this.designation = designation;
}

public String getDepartment() {
    return department;
}

public void setDepartment(String department) {
    this.department = department;
}

// Optional: Override toString method for easy display
@Override
public String toString() {
    return "Employee{" +
            "empID=" + empID +
            ", name='" + name + '\'' +
```

```
            ", salary=" + salary +
            ", designation='" + designation + '\'' +
            ", department='" + department + '\'' +
            '}';
    }
}
```

Assignment: 7
Aim:- Build a command-line utility using Node.js that performs a specific task, such as converting text to uppercase, calculating the factorial of a number, or generating random passwords.

```
#!/usr/bin/env node

const { program } = require('commander');
const crypto = require('crypto');

// Utility function to generate a random password
function generatePassword(length, useNumbers, useSpecialChars) {
    const alphabet =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
    const numbers = "0123456789";
    const specialChars = "!@#$%^&*()_+~`|}{[]:;?><,./-=";
```

```
    let characters = alphabet;
    if (useNumbers) characters += numbers;
    if (useSpecialChars) characters += specialChars;

    let password = '';
    for (let i = 0; i < length; i++) {
        const randomIndex = crypto.randomInt(0, characters.length);
        password += characters[randomIndex];
    }
    return password;
}

// Command-line options
program
    .version('1.0.0')
    .description('Generate a random password')
    .option('-l, --length <number>', 'length of password', '12')
    .option('-n, --numbers', 'include numbers', false)
    .option('-s, --special', 'include special characters', false)
    .parse(process.argv);

const options = program.opts();

// Generate password based on user input
const length = parseInt(options.length, 10);
const password = generatePassword(length, options.numbers, options.special);

console.log(`Generated Password: ${password}`);
```

Assignment: 8

Aim:- Develop a script that uses MongoDB's aggregation framework to perform operations like grouping, filtering, and sorting. For instance, aggregate user data to find the average age of users in different cities.

```
const { MongoClient } = require('mongodb');

async function runAggregation() {
    const uri = "mongodb://localhost:27017"; // MongoDB connection URI
    const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology:
true });

    try {
        await client.connect();
        console.log("Connected to MongoDB!");
```

```
        // Specify the database and collection
        const database = client.db("mydatabase");
        const usersCollection = database.collection("users");

        // Aggregation pipeline
        const pipeline = [
            {
                $group: {
                    _id: "$city",
                    averageAge: { $avg: "$age" },
                    userCount: { $sum: 1 }
                }
            },
            {
                $sort: { averageAge: -1 } // Sort cities by average age in descending order
            },
            {
                $match: { userCount: { $gte: 5 } } // Optional: Filter to show only cities with at
least 5 users
            }
        ];

        // Execute the aggregation
        const results = await usersCollection.aggregate(pipeline).toArray();

        console.log("Average age by city:");
        results.forEach(result => {
            console.log(`City: ${result._id}, Average Age: ${result.averageAge.toFixed(2)}, User
Count: ${result.userCount}`);
        });

    } catch (err) {
        console.error("Error running aggregation:", err);
    } finally {
        await client.close();
    }
}

runAggregation().catch(console.error);
```

Assignment: 9
Aim:- Assume four users user1, user2, user3 and user4 having the passwords pwd1, pwd2, pwd3 and pwd4 respectively. Write a servlet for doing the following: 1. Create a Cookie and add these four user id's and passwords to this Cookie. 2. Read the user id and passwords entered in the Login form and authenticate with the values available in the cookies.

Login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
</head>
<body>
    <h2>Login Form</h2>
    <form action="LoginServlet" method="POST">
```

```
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required>
        <br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <br><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

LoginServlet.java

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Map;

@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {

    // Hardcoded users and passwords
    private static final Map<String, String> users = new HashMap<>();

    static {
        users.put("user1", "pwd1");
        users.put("user2", "pwd2");
        users.put("user3", "pwd3");
        users.put("user4", "pwd4");
    }

    @Override
    public void init() throws ServletException {
        // Creating a cookie with all user data
        StringBuilder cookieValue = new StringBuilder();
        for (Map.Entry<String, String> entry : users.entrySet()) {

cookieValue.append(entry.getKey()).append("=").append(entry.getValue()).append(";");
        }
        // Adding the cookie
        Cookie userCookie = new Cookie("userAuth", cookieValue.toString());
        userCookie.setMaxAge(60 * 60 * 24); // Cookie expires in one day
        userCookie.setHttpOnly(true); // Make it HTTP-only for security

        // Add the cookie when the servlet initializes
        getServletContext().setAttribute("userCookie", userCookie);
    }
```

```
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        // Get username and password from request
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Retrieve the stored cookie
        Cookie[] cookies = request.getCookies();
        Cookie userCookie = null;
        for (Cookie cookie : cookies) {
            if (cookie.getName().equals("userAuth")) {
                userCookie = cookie;
                break;
            }
        }

        // Check if cookie exists and authenticate the user
        boolean isAuthenticated = false;
        if (userCookie != null) {
            String[] usersData = userCookie.getValue().split(";");
            for (String userData : usersData) {
                String[] parts = userData.split("=");
                if (parts[0].equals(username) && parts[1].equals(password)) {
                    isAuthenticated = true;
                    break;
                }
            }
        }

        // Prepare response
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        if (isAuthenticated) {
            out.println("<h2>Login Successful</h2>");
            out.println("<p>Welcome, " + username + "!</p>");
        } else {
            out.println("<h2>Login Failed</h2>");
            out.println("<p>Invalid username or password.</p>");
        }
    }
}
```

Assignment: 10
Aim:- Create a table which should contain at least the following fields: name, password, email-id, phone number Write Servlet/JSP to connect to that database and extract data from the tables and display them. Insert the details of the users who register with the web site, whenever a new user clicks the submit button in the registration page.
Set up database :

```
CREATE DATABASE UserDB;

USE UserDB;

CREATE TABLE Users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone VARCHAR(15) NOT NULL
```

);

Register.jsp:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>User Registration</title>
</head>
<body>
   <h2>Register</h2>
   <form action="RegisterServlet" method="POST">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" required><br><br>

      <label for="password">Password:</label>
      <input type="password" id="password" name="password" required><br><br>

      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required><br><br>

      <label for="phone">Phone Number:</label>
      <input type="text" id="phone" name="phone" required><br><br>

      <input type="submit" value="Register">
   </form>
</body>
</html>
```

Registerservlet.java:

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;

@WebServlet("/RegisterServlet")
public class RegisterServlet extends HttpServlet {
   private static final String JDBC_URL = "jdbc:mysql://localhost:3306/UserDB";
   private static final String JDBC_USERNAME = "root";
   private static final String JDBC_PASSWORD = "yourpassword";

   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      // Get form data
      String name = request.getParameter("name");
      String password = request.getParameter("password");
```

```
        String email = request.getParameter("email");
        String phone = request.getParameter("phone");

        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        try {
            // Load JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Connect to the database
            conn = DriverManager.getConnection(JDBC_URL, JDBC_USERNAME,
JDBC_PASSWORD);

            // Insert new user into the database
            String insertSQL = "INSERT INTO Users (name, password, email, phone) VALUES
(?, ?, ?, ?)";
            pstmt = conn.prepareStatement(insertSQL);
            pstmt.setString(1, name);
            pstmt.setString(2, password);
            pstmt.setString(3, email);
            pstmt.setString(4, phone);
            pstmt.executeUpdate();

            // Retrieve all users from the database
            String selectSQL = "SELECT * FROM Users";
            pstmt = conn.prepareStatement(selectSQL);
            rs = pstmt.executeQuery();

            // Display all users
            out.println("<h2>Registered Users</h2>");
            out.println("<table
border='1'><tr><th>ID</th><th>Name</th><th>Email</th><th>Phone</th></tr>");

            while (rs.next()) {
                out.println("<tr>");
                out.println("<td>" + rs.getInt("id") + "</td>");
                out.println("<td>" + rs.getString("name") + "</td>");
                out.println("<td>" + rs.getString("email") + "</td>");
                out.println("<td>" + rs.getString("phone") + "</td>");
                out.println("</tr>");
            }
            out.println("</table>");

        } catch (Exception e) {
            out.println("Error: " + e.getMessage());
```

```
            e.printStackTrace();
        } finally {
            // Clean up database resources
            try { if (rs != null) rs.close(); } catch (SQLException e) { e.printStackTrace(); }
            try { if (pstmt != null) pstmt.close(); } catch (SQLException e) { e.printStackTrace(); }
            try { if (conn != null) conn.close(); } catch (SQLException e) { e.printStackTrace(); }
        }
    }
}
```