| Assignment Rubrics | Marks split up | Mark Score | Total Marks |
|---|---|---|---|
| Assignment-I<br>Q-1 | Form design | 4 | |
| | Button design | 4 | |
| | Features of HTML | 2 | |
| Q-2 | Form design | 4 | |
| | Validation | 4 | |
| | Layout | 2 | |
| Q-3 | Layout design | 5 | |
| | Css background | 5 | |
| Assignment-2<br>Q-4 | Form design | 4 | |
| | Layout design | 4 | |
| | Navigation menu | 2 | |
| Q-5 | Form design | 5 | |
| | css layout | 5 | |

**Course Code /Title: CSA4399 – Internet Programming**
**Programme : Computer Science and Engineering**

## ASSIGNMENT QUESTIONS

### SET -2

| | | | | |
|---|---|---|---|---|
| 1 | You are updating a legacy web application to take advantage of modern web standards. The project involves transitioning from an earlier HTML version to HTML5. The goal is to leverage HTML5's new features to improve web development practices and enhance the functionality of web forms. | 10 | CO1 | 2 |
| 2 | Designing a user registration form for a new web application. The form needs to capture essential information such as name, email, and a message from users. It is crucial to implement client-side validation to ensure that the data entered is accurate and complete before the form is submitted. | 10 | CO1 | 3 |
| 3 | Designing a website that needs to function effectively across various devices and screen sizes. The design should include different types of layouts such as fixed, fluid, and responsive. To achieve this, you need to use CSS techniques like Flexbox or Grid to ensure that the layout adapts well to different screen sizes and maintains a consistent user experience. | 10 | CO1 | 2 |
| 4 | To create a webpage that offers a seamless user experience across devices, responsive design principles must be applied. This involves using fluid grids, flexible images, and media queries to ensure the layout adjusts gracefully to different screen sizes. For instance, on a desktop, the webpage might display multiple columns with detailed content, while on a smartphone, the same content could stack vertically for easy scrolling. Navigation menus should transform into a hamburger icon on smaller screens, ensuring they remain accessible without taking up too much space. Additionally, touch-friendly elements and optimized images ensure fast loading times and a smooth experience on all devices. | 10 | CO1 | 3 |
| 5 | Given a scenario (e.g., creating a blog post, a product listing), design a webpage using appropriate elements, tables, lists, and images for optimal readability and user experience. Also, describe step-by-step the sequence of HTTP requests and responses that occur when a user accesses a webpage containing multiple resources (HTML, CSS, JavaScript, images). | 10 | CO1 | 3 |

# Assignment - 1

1. You are updating a legacy web application to take advantages of modern web standards. The project involves transiting from an earlier HTML version to HTML5.

When updating a legacy web application to HTML5, you can take advantage of several features & best practices that improve both development & user experience.

## Semantic Elements :-

Use Semantic elements to replace the non-semantic divs and spans with HTML5 Semantic tags like <header>, <footer>, <article>, <section>, <nav>, and <aside>. This improves the document structure, accessibility & SEO.

Example :-

```
<header>
  <h1> Website title </h1>
</header>

<nav>
  <ul>
  <li><a
```

```
<li><a
href = "#content"> contact </a></li>

</ul>
</nav>
<section>
<article>
  <h2> Article Title </h2>
  <p> Content goes here. </p>
</footer>;
```

2. Enhanced Form controls: utilize new input types such as email, date, url, tel, number and range to provide better user input validation & UI.

example:

```
html.
<form action = "/submit" method = "post">
<label for = "email"> Email : </label>
<input type = "email" id = "email" name = "email" required
<label for = "website"> website : </label>
<input type = "url" id = "website" name = "website">
<button type = "submit"> Submit </button>
</form>.
```

3. Multimedia Integration:

Native Audio & video: Replace flash or other plugins with native HTML5 elements for audio & video. This reduce dependency on third party plugins and improves performance and accessibility.

Example: HTML

```
<video controls>
  <source src="movie.mp4" type="video/mp4">
  Your browser does not support this video tag.
</video>
<audio controls>
  <source src="song.mp3" type="audio/mpeg">
  Your browser does not support the audio element
</audio>
```

4. Offline Capabilities and storage:

Local storage and sessionstorage use localstorage for persistent data storage across sessions and session storage for data storage within a single session.

Example: Javascript:

```
localStorage.setItem('username', 'Ashok');
sessionStorage.setItem('sessionID', '12345');
const username = localStorage.getItem('username');
const sessionID = sessionStorage.getItem('sessionID');
```

5. Responsive Design: Ensure the application is responsible by defining the viewport and using media queries to adjust for different screen sizes.

Example: HTML

```
<meta name='viewport' content='width
= device-width initial-scale=1.0'>
<style>
<body>{
    font-family: Arial,
```

```
body{
background-color:
lightblue;
    }
}
</styles>
```

6. Improved Accessibility: Enhance ...
roles, states, and properties to make dynamic con...

Example: html

```
<nav aria-label = "main navigation">
<ul>
<li><a href = "#home"> Home </a> </li>
<li><a href = "#services"> Services </a></li>
<li><a href = "#contact"> Contact </a></ul>
</nav>.
```

7. Progressive Enhancement & Graceful Degradation:

Enhance new features to implement new HTML5 features with fallbacks for older browsers that may not support them.

Example: html

```
<script src="modernizr.js"></script>
<script>
if (! modernizr.inputtypes.date) {
  older browsers
} </script>.
```

8. CSS3 Integration: Integrate CSS3 features such as transitions, animations and flexbox for more advanced styling and layout options without relying on Javascript.

Example: CSS

```
box{
  width: 200px
  height: 200px
  background: coral;
  transition: background-color 0.5s.
              ease;
}
.box:hover{
background-color: teal
}.
```

2. Designing a user registration form for a new web application. The form needs to capture essential information such as name, email and a message from users.

When designing a user registration form for a web application, it's essential to focus on both the user experience and data integrity.

Form Structure:

```
<form id="registrationForm" action="/submit" method="post">
<label for="name">Name:</label>
<input type="text" id="name" name="name" required
minlength="2" maxlength="50" pattern="[A-Za-z\s]+">
<label for="email">Email:</label>
<input type="email" id="email" name="email" required>
<button type="submit">Registration</button>
</form>.
```

Client Side Validation with JavaScript:

```
document.getElementById('registrationForm').addEventListener('submit', function(event)
{
  const name = document.getElementById('name');
  const email = document.getElementById('email');
  const message = document.getElementById('message');
```

```
if (!name.value.match(/^[A-Za-z\s]+$/)){
    name.focus();
    event.preventDefault();
}
if (!message.value.match(/^[A-Za-z\s]+$/)){
    message.focus();
    event.preventDefault();
} if (!email.value.match(/^[A-Za-z0-9]+$/)){
    email.focus();
    event.preventDefault();
});
```

User Experience enhancement:

* Add placeholders to guide the user to input
* provid additional information on how for complex fields.
* Error message to show error message
Refine the user types of smoother experience.

Security Consideration:

Password: if your form includes a password field, ensure passwords are hashed before storing them.

Recaptcha: consider adding reCAPTCHA to prevent spam registrations

CSRF: Implement Cross-site request forgery protection to secure form submissions.

3. Des

3. Designing a website that needs to function effectively across various devices and screen sizes. The design should include different types of layout such as fixed, fluid and responsive.

understanding layout types:-

Fixed layout: A fixed layout uses specific pixel values for width, meaning the layout remains the same regardless of the screen size.

Fluid layout: A fluid layout uses percentage-based widths, allowing the layout to adapt to the different screen sizes.

Responsive layout: A responsive layout combines both fixed and fluid layouts, along with media queries, to adapt to various screen sizes, ensuring a consistent user experience.

Using CSS Techniques:-

Flexbox: Flexbox is ideal for one-dimensional layouts, where you need to align items along a single axis.

Grid: CSS grid is perfect for two dimensional layouts where both rows and columns are essential. It provide more control over the layout.

Combining layout types:

Fixed header with fluid content:

CSS

```
body, html {
    margin: 0;
    padding: 0;
}
.header {
    width: 100%;
    position: fixed;
    top: 0;
    background-color: #333;
    color: white;
    padding: 10px 0;
    text-align: center;
}
.main content {
    margin-top: 60px; /* push content
below the header */ padding: 20 px;
width: 80%; /* Fluid width */
max-width: 1200px; /* cap the
width on large screens */
    margin-left: auto;
    margin-right: auto;
}
```

Final thought: By using flexbox & grid in combination with media queries, you can create a flexible, responsive website that provides a seamless user experience across a wide range of devices.

4. To creat
user e
principl
Fluid
rather
layou
creat
Flex
con
ad
fre
the
Ex