INDEX

EXPERIMENT NO.	DATE	NAME OF THE EXPERIMENT	PAGE NO.	REMARKS
1	28/08/23	Demo on basic html tags	1	10/10 Sri 25/9/23
2(a)	04/09/23	Design a static login page for Shopping cart application	2	10/10 Sri 25/9/23
2(b)	11/09/23	Design a static Registration page for Shopping Cart application.	5	10/10 Sri 25/9/23
3(a)	25/09/23	Design a responsive Login page using Bootstrap 5	10	9/10 Sri 9/10/23
3(b)	25/09/23	Design a responsive registration page using Bootstrap 5	12	10/10 Sri 9/10/23
2(c)	25/09/23	Design a responsive Catalog page using Grid layout.	16	9/10 Sri 9/10/23

INDEX

EXPERIMENT NO.	DATE	NAME OF THE EXPERIMENT	PAGE NO.	REMARKS
2(d)	09/10/23	Design a homepage for Cart application	20	9/10 Sri
2(c)	09/10/23	Design a Cart page for Shopping Cart application	23	10/10 Sri
1(a)	09/10/23	XML file for Student Management System	27	9/10 Sri
5 1(b)	16/10/23	internal & external DTD for Student management System	29	9/10 Sri
1(c)	6/11/23	Validate XML using XSD	31	10/10 Sri
4(a)	6/11/23	Fat arrow functions in js	34	9/10 Sri
4(b)	13/11/23	Destructuring	36	
4(c)	13/11/23	Generator Function	37	
7(a)	13/11/23	Node module : Http	38	10/10 Sri
7(b)	13/11/23	Node module : OS	39	
5(a)	20/11/23	Call backs	41	
5(b)	20/11/23	promises	43	
5(c)	20/11/23	closures	45	9/10 Sri
Add Exp-4	20/11/23	Regular expressions	48	

INDEX

EXPERIMENT NO.	DATE	NAME OF THE EXPERIMENT	PAGE NO.	REMARKS
6	27/11/23	Realtime database in firebase for student management system.	50	(15/15) Sign 18/12/23
8	04/11/23	Create Rest API and perform CRUD operations.	53	(15/15) Sign 18/12/23
9	11/12/23	Develop an express application to perform crud application on student data using postman	56	(20/20) Sign 18/12/23
10	11/12/23	Create authorized endpoints using JWT.	59	(10/10) Sign 18/12/23
	18/12/23	Create a Angular application that displays "Hello CSIT"	62	
	18/12/23	Create a Angular application that displays the Todo List.	64	

Aim :- Demonstrate basic html tags.

```
<!DOCTYPE html>
<html>
    <head>
        <title> BASIC TAGS </title>
    </head>
    <body>
        <div align="center">
            <h1> BASIC TAGS </h1>
            <b> BOLD Text </b><br>
            <i> ITALIC TEXT </i><br>
            <u> UNDERLINED </u><br>
            <strike> STRIKE THROUGH TEXT </strike><br>
            <big> Important Text </big><br>
            <small> SMALL TEXT </small><br>
            <center> CENTER </center><br>
            <p> This is a paragraph </p>
            <hr>
            <pre> Spaces are preserved </pre><br>
            <tt> type writer text </tt><br>
            2 <sup> 2 </sup> = 4 <br>
            H <sub> 2 </sub> <br>
        </div>
    </body>
</html>
```

Aim :- Design a static login page for a shopping cart application.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Login</title>
    </head>
    <style>
        form
        {
            border-radius: 5px;
            align-content: center;
        }
        h2
        {
            text-decoration: underline;
        }
    </style>
    </head>

    <body>
        <br>
        <h2 align="center">Login page</h2>
        <br>
        <form action="post" method="post" name="align"
              align="center">
```

```
<table align = "center"  cellspacing = "20" >
    <fieldset >
        <legend > Login Here </legend >
        <tr>
          <td>
             <label > Enter username : </label >
          </td>
          <td>
             <input type = "text" name = "username" >
          </td>
        </tr>
        <tr>
           <td>
             <label > Enter password : </label >
           </td>
           <td>
             <input type = "password" name = "password" >
           </td>
        </tr>
        <tr>
           <td>
              <input type = "submit" value = "SUBMIT" >
           </td>
           <td>
              <input type = "reset" value = "RESET" >
           </td>
        </tr>
```

```
</table>
</fieldset>
</form>
</body>
</html>
```

O/P :-

Aim :- Design a static Registration page for shopping cart application.

```
<!DOCTYPE html>
<html>
    <head>
        <title> Registration Form </title>
        <meta name = "viewport" content = "width = device-width",
                                          initial-scale=1.0" >

        <style>
            table
            {
                background-color : lightgray;
                border : 2px solid black;
                border-radius : 10px;
                max-width : 500px;
            }
            h1
            {
                text-decoration : underline;
                font-size : 2vw;
            }
            input : hover
            {
                background-color : black;
                color : white;
            }
        </style>
    </head>
```

```
<body>
    <h1 align="centre"> Registration form </h1>
    <form>
        <table align="centre" cellspacing="30">
            <tr>
                <td>
                    <label> FirstName : </label>
                </td>
                <td>
                    <input type="text" name="f1">
                </td>
            </tr>
            <tr>
                <td>
                    <label> LastName : </label>
                </td>
                <td> <input type="text" name="f2">
                </td>
            </tr>
            <tr>
                <td>
                    <label> Email : </label>
                </td>
                <td>
                    <input type="email" name="e1">
                </td>
            </tr>
```

```
<tr>
    <td>
        <label >Password : </label >
    </td>
    <td>
        <input type ="password " name = "p1" >
    </td>
</tr>

<tr>
    <td>
        <label > confirm password : </label >
    </td>
    <td>
        <input type = "password " name = "p2" >
    </td>
</tr>

<tr>
    <td>
        <label> Mobile : </label >
    </td>
    <td>
        <input type = "text" name = "m1" maxlength = "10">
    </td>
</tr>

<tr>
    <td>
        <label > Date of Birth : </label >
    </td>
    <td>
        <input type = "date" name = "d1" >
    </td>
</tr>
```

```
<tr>
    <td>
        <label> Gender : </label>
    </td>
    <td>
        <input type="radio" name="r1" value="male">Male
        <input type="radio" name="r1" value="Female">Female
    </td>
</tr>
<tr>
    <td>
        <label> State : </label>
    </td>
    <td>
        <select>
            <option> Select state : </option>
            <option> Telangana </option>
            <option> Punjab </option>
            <option> Tamilnadu </option>
        </select>
    </td>
</tr>
<tr>
    <td>
        <label> Pincode : </label>
    </td>
    <td>
        <input type="text" name="p1" maxLength="6">
    </td>
</tr>
```

```
<tr>
  <td>
      <label> Upload photo : </label>
  </td>
  <td>
      <input type = "file" name = "pic" >
  </td>
</tr>
<tr>
    <td colspan = "2" >
        <input type = "checkbox" name = "c1" checked >
          I agree to terms and Conditions.
    </td>
</tr>
<tr>
    <td>
        <input type = "submit" value = "SUBMIT" >
    </td>
    <td>
        <input type = "reset" value = "RESET" >
    </td>
</tr>
</table>
</form>
</body>
</html>
```

Aim:- Design a responsive Loginpage for shopping Cart application using Bootstrap 5 framework.

```
<!DOCTYPE html>
<html Lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width = device-width, initial-
                                   scale=1.0">

    (cdn links)

</head>
<body>
    <br>
        <h2 align="center">Login Page</h2>
    <br>
    <div class="container" style="max-width:600px">

<form action="" method="post" name="login" align="center"
 class="border border-3 border-primary bg-light p-5
                                   mx-auto">

    <div class="p-2">
        <label class="form-label">Enter username :
                                                </label>

        <input type="text" name="username" class="form-
                                                controls">

    </div>
```

```
<div class = "p-2" >
<label class = "form-label" > Enter password : </label >
<input type = "password" name = "password" class = form-controls" >
</div>

<div class = "p-2" >
<input type = "submit" value = "SUBMIT" class = "btn
btn-primary" >

<input type = "result" value = "RESET" class = "btn btn-
secondary" >
</div>
</form>
</div>
</body>
</html>
```

Aim :- Design a responsive registration page for Shopping Cart application using Bootstrap 5 framework.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Registration Form</title>
        <meta charset="Utf-8">
        <meta name="viewport" content="width=device-width,
                                       initial-scale=1.0">

        (CDNLinks)
    </head>
    <body>
      <div class="container">
        <h1 align="center" style="text-decoration: underline">
                                Registration Form</h1>
        <h3 align="right" Style="text-decoration: dotted">
                                21B81A3341</h3>

<form name="f1" action="" method="post" class="border border-4
border-dark bg-warning p-3 m-3" auto Style="max-width: 50%">

<div>
    <label class="form-label">First Name:</label>
    <input class="form-control" type="text" name="f2">

</div>

<div>
    <label class="form-label">Last Name:</label>
    <input class="form-control" type="text" name="f3">

</div>
```

```
<div>
    <label class="form-label">Email:</label>
    <input class="form-control" type="email" name="e1">
</div>

<div>
    <label class="form-label">Password:</label>
    <input class="form-control" type="password" name="e2">
</div>

<div>
    <label class="form-label">Confirm-password:</label>
    <input class="form-control" type="password" name="e3">
</div>

<div>
    <label class="form-label">Mobile:</label>
    <input class="form-control" type="text" name="m1" maxLength="10">
</div>

<div>
    <label class="form-label">Date Of Birth:</label>
    <input class="form-control" type="date" name="dob">
</div>

<div>
    <label class="form-check-label">Genders:</label>
</div>

<div>
    <input class="form-check-input" type="radio" name="r1" value="Male">
    Male</div>
```

```
<div>
    <input class="form-check-input" type="radio" name="r1"
     value="Female">Female</div>

<div>
    <label class="form-label">Address:</label>
</div>

<div>
    <textarea class="form-control" rows="3" cols="20">
    </textarea>
</div>

<div>
    <select class="form-select">
        <option>Select State:</option>
        <option>Telangana</option>
        <option>Punjab</option>
    </select>
</div>

<div>
    <label class="form-label">Pincode:</label>
    <input class="form-control" type="text" name="P1" maxLength="6">
</div>

<div>
    <label class="form-label">Upload photo:</label>
    <input class="form-control" type="file" name="pic">
</div>
```

```
<div class="form-check">
    <input class="form-check-input" type="checkbox" name="c1" checked>
    <label class="form-check-label"> I agree to the terms and conditions
    </label>
</div>

<div>
    <input type="submit" value="SUBMIT" class="btn btn-primary">
    <input type="reset" value="RESET" class="btn btn-info">
</div>
</form>
</body>
</html>
```

Aim :- Design a responsive Catalog page for a Shopping cart application using grid layout.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
                                       initial-scale=1.0">

        (CDN LINKS)

        <style>
            .gcontainer
            {
                display : grid;
                grid-template-columns : auto auto auto auto;
                gap : 20px;
            }
            .card
            {
                width : 300px;
                border : 10px solid black;
                padding : 5px;
                text-align : center;
            }
            .cardimg
            {
                max-width : 100%;
                min-width : 100%;
            }
```

```
.card h2
{
    color : blueviolet;
    background-color : beige;
     font-size : 2vw;
     text-align : center;
}
.card h3
{
     color : burlywood;
     background-color : aqua;
       font-size : 1.5vw;
       text-align : centre;
 }
.card button
{
      background-color : blanchedalmond;
          color : blue;
}
@media only screen and (max-width : 1000px)
{     .gcontainer {
              display : grid;
              grid-template-columns : auto auto;
                gap : 20px;
        }
 }
```

```
@media only screen and (max-width : 600px ){
        .gcontainer
        {
            display : grid;
            grid-template-columns : auto;
             gap : 20px;
        }
}
</style>
</head>
<body style = "background-color : beige" > Catalog
    <h1 align = "center" style = "text-decoration : underline" >
                Application </h1>
    <h3 align = "right" > 21B81A3341 </h3>
    <br>
    <br>
    <div class = "gcontainer" >
        <div class = "card" >
            <img src = "pics/javaif" >
            <h2> Jeans for Men </h2>
            <h3> Price RS : <del > 3500 </del> RS 2000 </h3 >
            <p> The Indian Garage </p >
            <button > Add to cart </button >
            <button > Wishlist </button>
        </div>

    <div class = 'card' >
            <img src = 'pics/p.jpeg' >
            <h2> Top </h2>
            <h3> Price : RS <del > 1000 </del RS 650 </h3 >
            <p> Medium and Large Cotton Designer top </p >
```

```
        <button> Add to cart </button>
        <button> Wishlist </button>
    </div>

    <div class="card">
        <img src="pics/k.jpeg">
        <h2> Children Frock </h2>
        <h3> Price : RS <del>1000</del> RS 800 </h3>
        <p> Soft cloth, short frock </p>
        <button> Add to cart </button>
        <button> Wishlist </button>
</div>

<div class="card">
        <img src="pics/w.avif">
        <h2> watch </h2>
        <h3> price : RS <del>5000</del> 3500 </h3>
        <p> Elegant pearl strap watch </p>
        <button> Add to cart </button>
        <button> Wishlist </button>

    </div>
    </div>
</body>
</html>
```

Aim:- Create a homepage for Shopping Cart application.

```
<!DOCTYPE html>
<head>
    <meta name="viewport" content="width = device-width,
                                   initial-scale = 1.0" >

    <title> HOME </title>

    <style>

        .container
        {       height :100%;
                display : grid;
                grid-template-rows : 30%, 180%  30%;
                grid-template-columns : 30%  auto;
        }

        header
        {       background-color : red;
                grid-area : 1/1/1/span 2;
                text-align : center;
        }

        nav
        {       background-color : Cyan;
                grid-area : 2/1/span 1/span 1 ;
        }

        article
        {       background-color : bisque;
                grid-area : 2/2/span 1/span 1;
        }       text-align : center;
```

```
footer
  {
      background-color : black;
       grid-area : 3/1/span 1/span 2;
          color : white;
          text-align : center;
  }

img
  {     float : left;
  }

ul li
  {     list-style-type : circle;
          padding : 20px;
    }
  footer a
  {
        color : white;
  }
</style>
</head>
<body>
    <div class = "container">
      <header>
          <img src = "shopping.png" width = "200px height : 500">
          <h2> Online Shopping </h1>
      </header>
      <nav> <ul>
          <li ><a href = "#" > LOGIN </a></li>
          <li><a href = "#" > CART </a></li>
```

```
        </ul>
    </nav>

    <article>
        <p> welcome to Online Shopping </p>

    </article>

    <footer>
        <p> @copyright: onlineshopping.com </p>
        <a href = 'mailto: onlineshopping@gmail.com'>
            Contact us </a>

    </footer>

</div>
</body>
</html>
```

09/10/23

2) **Aim:-** Create a Static cart page for a shopping cart application.

```
<!DOCTYPE html>
<head>
    <title>Cart</title>
    <meta name="viewport" content="width=device-width,
                                   initial-scale=1">

    <style>
        .container
        {
            display: grid;
            grid-template-columns: 25% 25% 25% auto auto
            grid-template-rows: 2fr 2fr 2fr 2fr 1fr 2fr
                                2fr 1fr

            border: 2px solid black;
            border-radius: 10px;
        }

        .headings
        {
            display: flex;
            justify-content: space-around;
            grid-area: 1/1/span 1/span 5;
        }

        .product
        {
            display: flex;
            grid-area: 2/1/span 1/span 5;
            justify-content: space-around;
        }
```

```
.product
{
    display : flex;
    grid-area : 3/1/span 1/span 5;
    justify-content : space-around;
}

.coupon
{
    display : flex;
    grid-area : 4/1/span 1/span 1;
    padding : 50px;
}

#update
{
    grid-area : 4/5/span 1/span 1;
    margin : 5px;
}

.details
{
    display : flex;
    flex-direction : column;
    grid-area : 6/4/span 2/span 1;
    background-color : lightgray;
    text-align : center;
}
```

```
#card
{
    grid-area: 5/5/span 1/span 1;
}

#proceed
{
    grid-area: 5/5/span 1/span 1;
    margin: 15px;
}
</style>
</head>
<body>
    <h2> Cart </h2>
    <div class="container">
        <div class="headings">
            <hy> product img </hy>
            <hy> Product info </hy>
            <hy> Price C$> $ </hy>
            <hy> Quantity </hy>
            <hy> Sub total </hy>
        </div>
        <div class="product1">
            <img src="jeans.jpeg" width="50" height="50">
            <p> Jeans for men </p>
            <p> 200 </p>
            <p> 2 </p>
            <p> $400 </p>
        </div>
```

```
<div class="products">
    <img src="tshirt.jpeg" width="50" height="50">
    <p> Tshirt for Men </p>
    <p> 100 </p>
    <p> 1 </p>
    <p> $100 </p>
</div>

<div class="coupon">
    <input type="text" placeholder="EnterCoupon">
    <button> SUBMIT </button>
</div>

<button id="update"> Update Cost </button>
<hy id="costid"> Cost Details </hy>

<div class="details">
    <p> Sub Total: </p>
    <p> Shipping Cost: </p>
    <p> Total : </p>
</div>

<div class="textfields">
    <input type="text" value="$500">
    <input type="text" value="$100">
    <input type="text" value="$600">
</div>
<button id="proceed"> proceed to checkout </button>

</div>
</body>
</html>
```

09/10/23

Aim :-

i) a) Create an XML file for a Student management System.

```
<? xml version="1.0" encoding="utf-8" ? >
<students-list>
    <student roll="1" >
        <name>
            <fname> tim </fname>
            <mname> david </mname>
            <lname> s </lname>
        </name>
        <gender> Male </gender>
        <dob> 1/10/2003 </dob>
        <branch> CSIT </branch>
        <year> 3rd year </year>
        <sem> 1 </sem>
        <percentage> 99 </percentage>
        <contact-info>
            <email> tim@gmail.com </email>
            <mobile> 9876543210 </mobile>
            <address> abcd </address>
        </contact-info>
        <courses-registered>
            <course code="33" >
                <cname> java </cname>
                <cmentor> veena </cmentor>
            </course>
        </courses-registered>
    </student>
```

1) Aim:

Validate, the XML File for Students Management System using DTD with external and internal DTD.

External DTD

```
<!ELEMENT Student-list (student+) >
<!ELEMENT Student (name, gender, dob, branch, year, sem, percentage,
                   contact-info, courses-registered) >

<!ELEMENT name (fname, mname, lname)
<!ELEMENT fname (#PCDATA) >
<!ELEMENT mname (#PCDATA) >
<!ELEMENT lname (#PCDATA) >
<!ELEMENT gender (#PCDATA) >
<!ELEMENT dob (#PCDATA) >
<!ELEMENT branch (#PCDATA) >
<!ELEMENT year (#PCDATA) >
<!ELEMENT Sem (#PCDATA) >
<!ELEMENT percentage (#PCDATA) >
<!ELEMENT contact-info (email, mobile, address) >
<!ELEMENT email (#PCDATA) >
<!ELEMENT mobile (#PCDATA) >
<!ELEMENT address (#PCDATA) >
<!ELEMENT course-registered (course+) >
<!ELEMENT Course (cname, cmentor) >
<!ELEMENT cname (#PCDATA) >
<!ELEMENT cmentor (#PCDATA) >
<!ATTLIST Student roll CDATA #REQUIRED >
<!ATTLIST Course Code CDATA #REQUIRED >
```

<? xml version = "1.0" encoding = "utf-8" >
<! DOCTYPE Student-list SYSTEM "student.dtd" >

[XML]

internal DTD

<? xml version="1.0" encoding = "UTF-8" >
<! DOCTYPE Student-list [

DTD

]>

[XML]

Validate XML file using XSD.

```
<xs:schema xmlns:"http://www.w3.org/2001/XMLSchema">
<xs:element name = "Student_list">
<xs:complexType>
    <xs:sequence>
        <xs:element name = "Student" maxOccurs = "unbounded">
        <xs:complexType>
            <xs:sequence>
                <xs:element name = "name">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="fname" type="xs:String"/>
                        <xs:element name="mname" type="xs:String"/>
                        <xs:element name="lname" type="xs:String"/>
                    </xs:sequence>
                </xs:complexType>
                </xs:element>
                <xs:element name="gender" type="xs:String"/>
                <xs:element name="dob" type="xs:date"/>
                <xs:element name="branch" type="xs:String"/>
                <xs:element name="year" type="xs:String"/>
                <xs:element name="Sem" type="xs:String"/>
                <xs:element name="percentage" type="xs:float"/>
                <xs:element name="Contact-info">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="email" type="xs:String"/>
                        <xs:element name="mobile" type="xs:long"/>
```

```
<xs:element name="address" type="xs:string" />
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="courses-registered">
<xs:complexType>
    <xs:sequence>
        <xs:element name="course" maxOccurs="unbounded">

            <xs:complexType>
                <xs:sequence>
                <xs:element name="cname" type="xs:String" />

                <xs:element name="Cmentor" type="xs:String" />

                <xs:sequence>
                <xs:attribute name="code" type="xs:int"
                use="required" />

            </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="sall" type="xs:int" use="required" />
</xs:complexType>
</xs:element>
```

```
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

O/P :-

The XML document Student.xml is valid.

Exploring ES - 6 Features

a) Fat arrows

```
Const f = () => "fat arrow function";
Console.Log ("no parameters:" + f());

Const a = (b) => b
Console.log (a(5));

Const mul = (a,b) => a*b;
Console.log ("mul:" + mul(1,2));

Const add (a,b,c) => a+b+c;
Console.log ("add:" + add(1,2,3));

Const ad = (...a) =>
{     Let sum = 0;
      for(i=0; i<a.length; i++)
      {     sum = sum + a[i];
      }
      return sum;
}
Console.Log ("multiple parameters:" + ad(1,2,3,4,5));
```

O/P :-

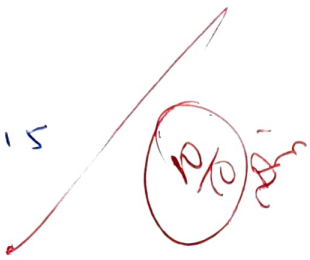
No parameters : for arrow function

5

mul : 2

Add : 6

Multiple parameters : 15



b) Destructuring

```
Const a = [1, 2, 3, 4, 5];

Const [x, y] = a;

Console.log ("x: " + x + " " + "y: " + y);

function cal (c, d)

{    let sum = c + d;

     let pro = c * d;

       return [sum, pro];

}

Const [sum, pro] = cal [x, y);

Console.log ("sum: " + sum + " " + product: " + pro);
```

O/P :-

x : 1 y : 2

Sum : 3 product : 2

c) Function generators

```
function * gen()
{
    Console.log("this statement is executed first");
    yield "hello";
    Console.log("this statement is executed after a pause");
    yield 10;
}
let obj = gen();
Console.log(obj.next());
Console.log(obj.next());
Console.log(obj.next());
```

o/p :-

this statement is executed first

{value: "hello", done: false}

this statement is executed after a pause

{value: 10, done: false}

{value: undefined, done: true}

Exploring various node modules

a) http

```
let hp = require('http');
let port = 5000;
let host = 'localhost';
let serve = hp.createServer(function(req,res)
{
    res.write("Ch.Sathvik (21B81A3341)<br>")
    if(req.url=='/')
    {
        res.write("page1");
        res.end();
    }
    else if(req.url=='/page2')
    {
        res.write("page2");
        res.end();
    }
});
Server.listen(port, host, ()=>{console.log('Server started')});
```

b) os

```
let os = require('os');
console.log("Name of os:" + os.type());
console.log("CPU Architecture:" + os.arch());
console.log("Total Memory:" + os.totalmem());
console.log("Free Memory:" + os.freemem());
console.log("User Information:" + os.userInfo());
let c = os.cpus();
console.log("Number of cpus=" + c.length());
for(i in c)
{   console.log(c[i]);
}
```

O/P:-

```
Name of os: Windows_NT
CPU Architecture : x64
Total Memory : 16518639616
Free Memory : 8224768000
User Information : [object object]
Number of cpus: 12
{
    model: 'AMD Ryzen 5 5500U with Radeon Graphics',
    Speed: 2096,
    times : { user : 1888468,
              nice : 0
              sys: 991056
```

idle : 2807722106

irq : 958703

},

{

model: 'AMD Ryzen 5 5500U with Radeon graphics

Speed: 2096,

times :

{ user: 431 2046,

nice : 0,

Sys : 482765,

idle : 3049 27640,

irq : 33937

}

}

Aim:- Explore the features of ES6

(1) Callbacks

```
function Sum(x, y, display)
{
        var z = x + y;
        display(z);
}

function display(res)
{
        console.log("result : " + res);
}

Sum(10, 5, display)
```

b) Callbacks

```
function callback()
{
    console.log("callback executed");
}
function a()
{
    console.log("function before callback");
}
function b()
{
    console.log("function after callback");
}
a();
setTimeout(callback, 3000);
b();
```

(b) promises ~~Callback function~~ using openweather API

```
<!DOCTYPE html>
<head>
    <Script>
        function getdata()
        {
        var city = document.getElementById('ll').value;
        var key = '76..'
        var url = 'https://api.openweathermap.org/data/2.5/
                  weather?q=${city}&appid=${key}'
          let p = fetch(url)
        p.then(function(response)
        {
            return response.json();
        }).then(function(data)
        {
            console.log(data);

let t = 'Temperature:' + data.main.temp + 'kelvin';
let h = '<br>Humidity:' + data.main.humidity;
let desc = '<br>Description:' + data.weather[0].description;
document.getElementById('l').InnerHTML = t+h+desc; });.
```

```
        p.catch ( function (error)
        {     console.Log ("error");
        })
    }
    </Script>
</head>
<body>
        <input type = "text" id = 'll' >
    <input type = "button" onclick = "getdata()" value = "info" >
        <h2 id = 'l' ></h2>
</body>
</html>
```

Aim :- Exploring features of ES6

(i) closures

```
<script>
    function outer()
    {   document.write("Outer function <br>");
        function inner()
        {   document.write("inner function");
        }
        inner()
    }
    outer();
</script>
```

O/P :-

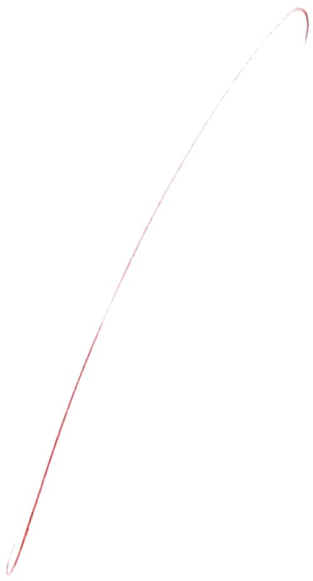
outer function
inner function

(ii) closures

```
<script>
    function outer()
    { document.write("outer function <br>");

        function inner()
        {
            document.write("inner function");

        }
      return inner;
    }
    var x = outer();
    Console.log(x);
    x()
</script>
```



O/P :-

outer function
inner function

Explore the features of ES6

(i) promises

```
<script>
var mypromise = new promise ((resolve, reject) =>
{   var done = true;
      if (done)
       {   resolve ("done");
       }
      else
       {   reject ("not done");
       }
});
mypromise.then (done) =>
{   document.write (done);
});
mypromise.catch (error) =>
{   document.write (error);
});
</script>
```

o/p:-

done

Aim: write a program to validate the field of login form using Regular expressions

```
<html>
<head>
<script>
function validate()
{
    var u = document.Login.t1.value;
    var p = document.Login.t2.value;
    var uexp = /^[a-z A-Z _ ]+$/;
if (u=="")
 {
    alert ("Enter username");
    return false;
 }
else if (p=="")
{
    alert ("Enter password");
    return false;
}
else if (uexp.test(u) == false)
 {
    alert ("username should contain alphabets and _");
    return false;
 }
else if (p.length <6)
{
    alert ("minimum should be 6 characters");
    return false;
}
```

```
else
{   alert("validate successful");
    return true;

 }

}
</Saipt>
</head>
<body>
  <form name="login" action="Success.html" method="post" onSubmit=
    "return validate()">

  Username : <input type="text" name="t1" >
  <br>
   password : <input type="password" name="t2" >
  <br>
   <input type="Submit" value="submit" >

  </form>
  </body>
  </html>
```

Aim :- Create a real time database in firebase for the student management system & explore the features of firebase realtime database.

```
<html>
<body>
    <input type="text" placeholder="Student roll no" id="roll"><br>
    <input type="text" placeholder="Student name" id="name"><br>
    <input type="text" placeholder="Student branch" id="br"><br>
    <input type="text" placeholder="Student cgpa" id="cgpa"><br>

<button id="b1">INSERT</button>
<button id="b2">UPDATE</button>
<button id="b3">DELETE</button>
<button id="b4">DISPLAY</button>

<h2 id="h"></h2>

<script type="module">
    document.getElementById("b1").addEventListener("click", insertData);
    document.getElementById("b2").addEventListener("click",
                                                   updateData);
    document.getElementById("b3").addEventListener("click",
                                                   deleteData);
    document.getElementById("b4").addEventListener("click",
                                                   displayData);

Const dbref = getDatabase();
```

```
function insertData()
{
    var r = document.getElementById("roll").value;
    var n = document.getElementById("name").value;
    var b = document.getElementById("br").value;
    var y = document.getElementById("year").value;
    var c = document.getElementById("cgpa").value;

    var obj = {
                name : n,
                branch : b,
                year : y,
                cgpa : c
              }
    set(ref(dbref, "SMS/" + r), obj);
    alert("Student Record inserted");
}

function displayData()
{
    var r = document.getElementById("roll").value;
    get(child(ref(dbref, "SMS/" + r))).then(function(data)
    {
        if(data.exists())
        {
            var ob = data.val();
            document.getElementById("h").innerHTML =
                ob.name + " " + ob.branch + " " + ob.year + " " +
                ob.cgpa;
        }
    }
```

```
        else
        {
            alert("No Data available");
        }
    });
}
function deleteData()
{
    var r = document.getElementById("roll").value;
    remove(ref(dbref, "SMS/" + r));
    alert("Student record deleted");
}
function updateData()
{
    var r = document.getElementById("roll").value;
    var n = document.getElementById("name").value;
    var b = document.getElementById("br").value;
    var y = document.getElementById("cgpa").value;
    var ob = {
        name: n,
        branch: b,
        year : y,
        cgpa : c
    }
    update(ref(dbref, "SMS/" + r), ob);
    alert("Student record updated");
}
</Script>
</body>
</html>
```

Aim Create Rest API and perform CRUD operations.

```
var http = require('http');
var url = require('url');
var fs = require('fs');
var stu = require('./student.json');
var port = 7000;
var serve = http.createServer((req, res) =>
{
    if (req.url === "/")
    {
        res.write("<h1> Welcome to API </h1>");
        res.end();
    }
    if (req.url === "/student" && req.method === "GET")
    {
        fs.readFile("student.json", function (err, data)
        {
            res.write(data);
            res.end();
        });
    }
    if (req.method === 'POST')
    {
        var newstu = url.parse(req.url, true).query;
        stu.push(newstu);
        writeDataBack(stu);
```

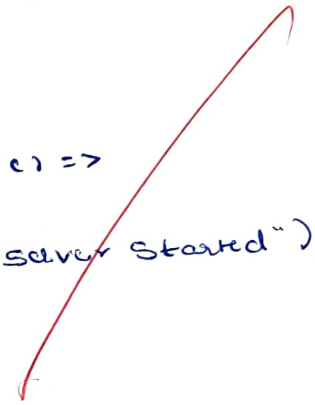
```
    res.write ("<h1> new Student record added Successfully </h1>");
    res.end();
}
if (req.method == "PUT")
{   var upstu = url.parse (req.url, true).query;
    for (s in stu)
    {   if (stu[s]['id'] == upstu['id'])
        {   stu[s]['name'] = upstu['name'];
            stu[s]['year'] = upstu['year'];
            stu[s]['cgpa'] = upstu['cgpa'];
            writeDataBack (stu);
            res.write ("<h1> New Student record updated Successfully
                </h1>");
        }
    }
    res.end();
}
if (req.method == "DELETE")
{   var delstu = url.parse (req, url, true).query;
    for (s in stu
    {   if (stu[s]['id'] == delstu['id'])
        {   stu.splice (s, 1);
            writeDataBackup (stu);
            res.write ("<h1> New Student record deleted
                Successfully </h1>");
        }
```


Vastunagar, Mangalpalli (V) Ibrahimpatan (M), R.R. Dist. Ph - 501 510

```
        res.end();
    }

function writeDataBase(data){
    fs.writeFile("Student.json", JSON.stringify(data),
        function(err){
            console.log(err);
        });
    }
});

server.listen(port, () =>
{   console.log("server started");
});
```



Aim :- Develop an express application to perform Crud application on student data using postman.

Commands for installing dependencies.

Express :- npm i express

body-parser - npm i body-parser.

```
Const express =require ('express');
const  app> = express();
const  bp = require ('body-parser');
Const  fs = require ('fs');
Const  stu = require ('./Students.json');
const  port = 7050;

app.use (bp.urlencoded ({extended: true}));
app.use (bp.json());

app.get ("/", function (req, res)=>

{
        res.send ("welcome to my API");
        res.end();

});

app.get ("/Students", function (req,res)

{
        fs.readFile ("Students.json", function (err,data)
        {       res.send (data);
                res.end();

        });

});
```


Vastunagar, Mangalpalli (V) Ibrahimpatan (M), R.R. Dist. Ph - 501 510

```
app.post ('/students', function (req, res)
{       var newstu = req.body;
        stu.push (newstu);
        writeDataBack (stu);
        res.send ("<h1> New Student Record Added Successfully
                                                <h1>");
        res.end();
});

app.put ("/students", function (req, res)
{   for (i in student)
    {   if (i in student)
        {   student[i]["name"] = req.body.name;
            student[i]['cgpa'] = req.body.cgpa;
            writeDataBack (student);
            res.send ("updated");
        }
    }
    res.end();
});

app.delete ("/students", (req, res) =>
{   for (i in student)
    {   if (student[i]["id"] = req.body.id)
        {   student.splice (i, 1);
```


Vastunagar, Mangalpalli (V) Ibrahimpatan (M), R.R. Dist. Ph - 501 510

```
                writeDataBack(student);
                res.send("deleted");
            }
        }
        res.end();
});

function writeDataBack(data)
{
    fs.writeFile("students.json", JSON.stringify(data, err) =>
    {
        console.log(err);
    });
}

app.listen(port, () =>
{
    console.log("Server started");
});
```

Aim:- Create authorized endpoints using JWT.

dependencies

npm i express

npm i jsonwebtoken

```
Const express = require('express');
Const app = express();
Const jwt = require('jsonwebtoken');
const port = 7060;
Const fs = require('fs');
Const key = "secret";
Const dbuser = {
                uname : "csit",
                password : "12345"
        });

app.post("/", function (req, res)
{
        var name = req.query.name;
        var pwd = req.query.password;
        if (dbuser.uname == name && dbuser.password == pwd)
        {
                jwt.sign ({dbuser}, key, function (req, res)
                {       res.json ({token});
                        res.end();
                });
        }
```

```
    else
      {
          res.send("invalid credentials");
          res.end();
      }
  }
app.get("/Students", validate, function(req,res)
{     fs.readFile("./Students.json", function(err,data)
      {       res.write(data);
              res.end();
      });
});

function validate(req,res,next)
{
      var auth = req.headers.authorization;
      var barray = auth.split(" ");
      var token = barray[1];

    jwt.verify(token,key,function(err,payload)
    {   if(err)
          {   res.send("invalid JWT token");
              res.end();
          }
        else
          {   next();
          }
    });
});
```

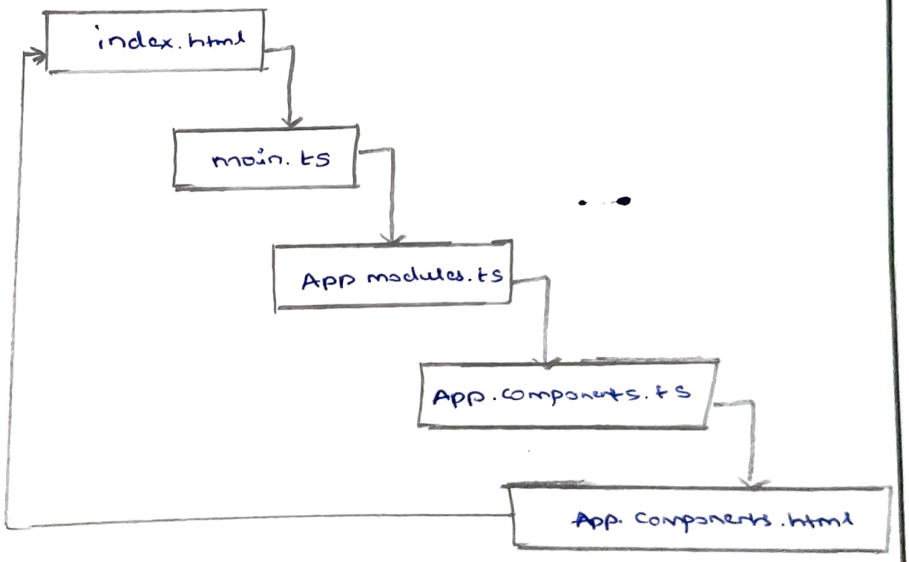
```
app.listen(port, () =>
{       console.log("Server started");
});
```

Aim :- Create an angular application to display "Hello CSIT". Sketch the project workflow and list various CLI commands used to create and run application.

CLI Commands

```
ng new my-app --no --standalone
ng serve
```

Angular Application workflow



app.component.ts

```
import {{component}} from '@angular/core';

@Component ({
        selector : 'app-root',
        templateUrl : './app.component.html',
        styleUrl : './app.component.css'
})
```

```
export class AppComponent {
    msg = "Hello CSIT";
}
```

app.component.html

```
<h1> msg from root component : {{msg}} </h1>
```

Aim :- Create an angular application with necessary components that displays TODO List and list various CLI commands used to create and run the application.

CLI commands

```
ng new project -no-standalone
ng Serve
```

app.component.ts

```
import {component} from '@angular/core';
@Component({
        selector : 'app-root',
        templateUrl : './app.component.html',
        styleUrl : './app.component.css'
})
export class AppComponent
{
    list = ["watching TV","playing basketball","Complete assignment writing"]
},
```

app.component.html

```
<h1> TODO List : </h1>
<h2>
    <div *ngIf = " List.length >0 then b1 ; else b2 "></div>
        <ng-template #b1>
            <ul *ng for = "let c of List" >
```

```
            <li> {{c}} </li>
        </ul>
        </ng-template>
        <ng-template #ba>
            <h1> No items </h1>
        </ng-template>
</h2>
```