



**Program: Diploma in information Technology**

---

**A**  
**Laboratory Manual**

**Emerging Trends in IT**  
**(Course Code EIT190912)**

**Semester: VI**

## **Vision /Mission Statements**

### **Institute Vision**

**SBM Polytechnic aspires to be the lead institute in providing need based technical education.**

### **Institute Mission**

- **To provide state-of-the-art infrastructure and latest equipments for providing a stimulating learning environment.**
- **To prepare students to meet the dynamic needs of the industry by periodic reviewing and upgradation of curriculum through an interactive process with industry.**
- **To inculcate a spirit of excellence in terms of academic performance, research and innovation in faculty by providing appropriate support and incentive systems.**
- **To promote and support Co-Curricular, extra-curricular activities and industry interaction to make students socially sensitive and employable.**

### **Department Vision**

**To create a learning environment that nurtures students and transform them into competent IT Diploma Graduates.**

### **Department Mission**

**M1: To impart technical and managerial skills for pursuing academic excellence through dynamic learning environment.**

**M2: To foster industry ready graduates by acquiring and utilizing latest technology.**

**M3: To strengthen holistic development and Professionalism in the Diploma graduates.**

## **PROGRAMME EDUCATIONAL OBJECTIVES**

**PEO1: Imbibe core knowledge and utilise associated technologies to provide domain related solutions.**

**PEO2: Be capable of adapting to the rapid pace of technological dynamics through professional competence.**

**PEO3: Develop a holistic personality equipped with leadership qualities, team skills and humane approach towards society.**

### **Program Outcomes**

1. **Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems**
2. **Problem analysis: Identify and analyse well-defined engineering problems using codified standard methods**
3. **Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs**
4. **Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements**
5. **Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.**

6. Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities
7. Life-long learning: Ability to analyse individual needs and engage in updating in the context of technological changes

### Program Specific Outcomes

#### Course Outcomes

PSO1: Students will demonstrate fundamental knowledge in core domains of IT such as Software Development, Databases and Information Systems

PSO2: Students will acquire skills that can provide IT solutions in the field of Networking, IOT, Machine Learning and Cloud Computing.

#### At the end of the semester student will be able to: -

- 1 Interpret basic knowledge on the working of various semi-conductor devices
- 2 Use diode in various electronic circuits
- 3 Use transistor for different switching circuits
- 4 Demonstrate the concepts of amplifier and photoelectric devices

#### CO-PO Mapping

Course Outcomes (CO)	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PSO 1	PSO 2
CO 1	1	2					1	3	
CO 2	3	2		2				3	
CO 3	3	2		3					1
CO 4	3						2		2
AVG	2.5	2		2.5			1.5	3	1.5

**LIST OF EXPERIMENTS**

<b>Sr. No.</b>	<b>Name of Experiment/ Assignment/ Sheet/ Job</b>	<b>Approx Hrs Required</b>	<b>CO</b>
1.	Create a basic responsive site using meta viewport tag and Media Queries, using CSS properties.	2	CO1
2.	Create a very basic custom float-based grid system. Understanding container, row, cols, margin, gutter and clearfix. Using Bootstrap v3 grid system (col-xs, col-sm, col-md, col-lg).	2	CO1
3.	Creating a very basic custom grid system will be flex-based. Using Bootstrap v4 grid system.	2	CO1
4.	Download and install Node.js, Command Line Interface	4	CO2
5.	Create first Application using Node.js	2	CO2
6.	Creating and using custom components and using the <u>styled-components</u> library for adding CSS to React Components.	4	CO2
7.	Use react hooks: useState and useRef.	2	CO2
8.	Create a basic todo app in ReactJS and using concepts as needed. (mini project)	4	CO2
9.	Use GitHub Actions to create custom build workflow.	2	CO3
10.	Install and Configure Splunk, Upload sample data using Splunk web and Execute search commands in Splunk web.	4	CO4
11.	Prepare and present case study on anyone application of Data Science	2	CO4
12.	Prepare and present case study on anyone application of Artificial Intelligence	2	CO4
	Total	32	

## **Experiment: - 1**

**Aim:-** Create a basic responsive site using meta viewport tag and Media Queries, using CSS properties.

**Lab outcome:-** Students will be able to set the viewport meta tag to control the width and scaling of the viewport to it's sized correctly on all devices.

### **Theory: -**

#### **Responsive web design**

- Responsive web design is about creating web pages that look good on all devices.
- Responsive Web Design uses using HTML and CSS.
- Responsive Web Design is not a program or a JavaScript.
- It is called responsive web design when you use css and html to resize, hide, shrink, or enlarge, a website, to make it look good on any screen.

#### **What is The Viewport?**

- The viewport is the user's visible area of a web page.
- The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.
- Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.
- Then, when we started surfing the internet using tablets and mobile phones, fixed size web pages were too large to fit the viewport. To fix this, browsers on those devices scaled down the entire web page to fit the screen.

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

*Syntax - <meta name="viewport" content="width=device-width, initial-scale=1.0">*

- The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device). The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

## What is a Grid-View?

- Many web pages are based on a grid-view, which means that the page is divided into columns.
- Using a grid-view is very helpful when designing web pages. It makes it easier to place elements on the page.
- A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.
- To build a responsive grid view ensure that all HTML elements have the box-sizing property set to border-box. This makes sure that the padding and border are included in the total width and height of the elements.

## What is a Media Query?

- Media query is a CSS technique introduced in CSS3.
- It uses the @media rule to include a block of CSS properties only if a certain condition is true.
- Example: If the browser window is 600px or smaller, the background color will be lightblue:

```
@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

## CODE:

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

```
    box-sizing: border-box;
}
.row::after {
    content: "";
    clear: both;
    display: block;
}
[class*="col-"] {
    float: left;
    padding: 15px;
}
html {
    font-family: "Lucida Sans", sans-serif;
}
.header {
    background-color: #9933cc;
    color: #ffffff;
    padding: 15px;
}
.menu ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}
.menu li {
    padding: 8px;
    margin-bottom: 7px;
    background-color: #33b5e5;
    color: #ffffff;
```

```
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}
.menu li:hover {
    background-color: #0099cc;
}
.aside {
    background-color: #33b5e5;
    padding: 15px;
    color: #ffffff;
    text-align: center;
    font-size: 14px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}
.footer {
    background-color: #0099cc;
    color: #ffffff;
    text-align: center;
    font-size: 12px;
    padding: 15px;
}
/* For desktop: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
```



```
.col-9 { width: 75%;}
.col-10 { width: 83.33%;}
.col-11 { width: 91.66%;}
.col-12 { width: 100%;}
@media only screen and (max-width: 768px) {
  /* For mobile phones: */
  [class*="col-"] {
    width: 100%;
  }
}
</style>
</head>
<body>
<div class="header">
  <h1>Chania</h1>
</div>

<div class="row">
  <div class="col-3 menu">
    <ul>
      <li>The Flight</li>
      <li>The City</li>
      <li>The Island</li>
      <li>The Food</li>
    </ul>
  </div>
  <div class="col-6">
    <h1>The City</h1>
```

<p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.</p>

</div>

</body>

</html>

**Conclusion:-** Performed Setting the viewport meta tag lets we control the width and scaling of the viewport so that it's sized correctly on all devices.

**Questions:-**

1. What is View port
2. What is Grid View
3. What id Media Query

## **Experiment: - 2**

**Aim:** - To create basic custom float-based grid system and understand container, rows,

columns, margins, gutter and clear-fix, using Bootstraps v3 grid system (col-x, col-sm, col-md, col-lg).

**Lab Outcome:** Students will be able to use a responsive grid to allow a layout to change dynamically based on the size of the screen.

### **Theory: -**

#### **BOOTSTRAP GRID SYSTEM**

The Bootstrap Grid System allows up to 12 individual columns across the page. If you do not want to use all 12 columns you can group the columns together to create wider columns. The Bootstrap Grid System is responsive and the columns will rearrange depending on the screen size. On a big screen it might look better with the content organized in three columns, but on a small screen it would be better if the content items were stacked on top of each other.

#### **GRID CLASS**

*The Bootstrap Grid System has 4 classes: -*

- 1.) X5 (for phone screens less than 768 pixels wide).
- 2.) 5m (for tablet screens equal to or greater than 768 pixels wide).
- 3.) md (for small laptop screens equal to or greater than 992 pixels wide).
- 4.) lg (for laptop and desktop screens equal to or greater than 1200 pixels wide).

The classes above can be combined to create more dynamic and flexible layouts.

#### **GRID SYSTEM RULES**

- Rows must be placed within a '.container' class (fixed width) or .container-fluid (full-width) for proper alignment and padding.
- Use rows to create horizontal group of classes.
- Content should be placed within columns, and only columns may be immediate children of rows.
- Pre-defined classes like .row and .col-sm-4 are available for quickly making grid layouts.

- Columns create gutter (gaps between column content) via padding. That padding is offset in rows for the first and last columns via negative margin on rows.
- Grid columns are created by specifying the number of available columns you wish to span. For e.g., three equal columns would use three (.col-sm-4) property.
- Column width is given in percentage, so they are always fluid and size relative to their parent element.

*E.g., Source Code: -*

```
<html>
<head>
<title> Bootstrap Example </title>
<meta charset = “utf-8”>
<meta name = “viewport” content = “width = device-width, initial-scale = 1”>
<link rel = “stylesheet”href=“https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/
css/bootstrap.min.css”>
<script src = “https://ajax.googleapis.com/ajax/libs/jquery/min.js”>
</script>
</head>
<body>
  <div class = “container-fluid”>
    <h1> Hello World !!! </h1>
    <p> Resize browser window to see the effect </p>
    <p> The columns will automatically stack on top of each other when the
screen is less
    than 768 px wide </p>
  <div class = “row”>
    <div class = “col-sm-4” style = “background:color = red”>
      (col-sm-4-1) </div class>
    <div class = “col-sm-4” style = “background:color = blue”>
```

```
(col-sm-4-2) </div class>  
<div class = "col-sm-4" style = "background:color = green">  
(col-sm-4-3) </div class>  
</div></div>  
</body>  
</html>
```

**Conclusion:-** Performed to create basic custom float-based grid system and understand container, rows, columns, margins, gutter and clear-fix, using Bootstraps v3 grid system

**Questions:-**

1. What is Bootstrap Grid system
2. What is Grid class and Grid system rules

### **EXPERIMENT NO :- 3**

**AIM:-** Creating a very custom based grid system will be flex based . Using Bootstrap v4 grid system

**Lab Outcome:** Students will able to use Bootstrap frame work to develop web page with help of grid sytem.

#### **THEORY:-**

**Bootstrap:-** Bootstrap is a free and open source web development framework . It is designed to ease the web development process of responsive, mobile-first websites by providing a collection of syntax for template designs .

It ensures all interface element of a website work optimally on all screen sizes.

**Bootstrap Version:-**

- Bootstrap first version was bootstrap 1.0.0.
- The next version was bootstrap 3 , bootstrap 4 , and the latest version is bootstrap 5.
- Bootstrap 4 was released on 7<sup>th</sup> January 2018
- Bootstrap 5 was released on 7<sup>th</sup> December 2020
- Bootstrap 4 has better and improvised options to create your website easily and quickly

**Compare bootstrap 3, bootstrap 4, bootstrap 5:-**

BOOTSTRAP 3	BOOTSTRAP 4	BOOTSTRAP 5
No support for I.E	No support for I.E 8 and 9 only	Support internet explorer 8 and 9
Separates both portrait and landscape mode on smartphone	Separates both portrait and landscape mode on smartphone	Both portrait and landscape mode looks identical
Container size 1330 px	Container size 1140 px	Container size 1170 px
Six class prefixes:- (col-xxl, col-xl, col-lg, col-md, col-sm, col-)	Four class prefixes:- (col-xl, col-md, col-sm, col-)	Four class prefixes:- (col-lg , col-mg, col-sm, col-xs)
Use CSS Flexbox, CSS Functions and variables	Use CSS Flexbox	Use CSS Float and clear
Only responsive layout	Only responsive layout	Non responsive layout

		option
Uses better cards	Uses cards instead of users and panels	No cards

**Flex-box:-** The flex-box layout module . makes it easier to design flexible responsive layout structure without using float or positioning.

**Properties:-**

Property	Descriptions
Align-self	Specifies the alignment for a flex item (overrides the flex container's align-items property)
Flex	A shorthand property for the flex-grow, flex-shrink, and the flex-basis properties
Flex-basis	Specifies the initial length of a flex
Flex-grow	Specifies how much a flex item will grow relative to the rest of the flex items inside the same container
Flex-shrink	The flex-shrink property specifies how much of a flex-item will shrink relative to the rest of the flex item
order	Specifies the order of the flex items inside the same container

**Example:-** To create a flexbox container and to transform direct children into flex items, use the d-flex class

```
<div class="d-flex p-3 bg-secondary text-white">
  <div class="p-2 bg-info">Flex item 1</div>
  <div class="p-2 bg-warning">Flex item 2</div>
  <div class="p-2 bg-primary">Flex item 3</div>
</div>
```

**Output:-**



Code:-

```
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
  <script
src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
<div class="container mt-3">
  <h2>Horizontal Direction</h2>
  <p>Use .flex-row to make the flex items appear side by side (default):</p>
  <div class="d-flex flex-row bg-secondary mb-3">
    <div class="p-2 bg-info">Flex item 1</div>
    <div class="p-2 bg-warning">Flex item 2</div>
    <div class="p-2 bg-primary">Flex item 3</div>
  </div>
  <p>Use .flex-row-reverse to right-align the direction:</p>
```



```
<div class="d-flex flex-row-reverse bg-secondary">
  <div class="p-2 bg-info">Flex item 1</div>
  <div class="p-2 bg-warning">Flex item 2</div>
  <div class="p-2 bg-primary">Flex item 3</div>
</div>
</div>
</body>
</html>
```

Output:-

## Horizontal Direction

Use .flex-row to make the flex items appear side by side (default):



Use .flex-row-reverse to right-align the direction:



**Conclusion:-** Performed creating a very custom based grid system will be flex based . Using Bootstrap v4 grid system.

**Questions: -**

1. Compare Bootstrap Versions
2. What is Flex-box

## **Experiment No.: 4**

**Aim:** Download and install Node.js, Command Line Interface

**Lab Outcome:** Students will able to download and install Node.js

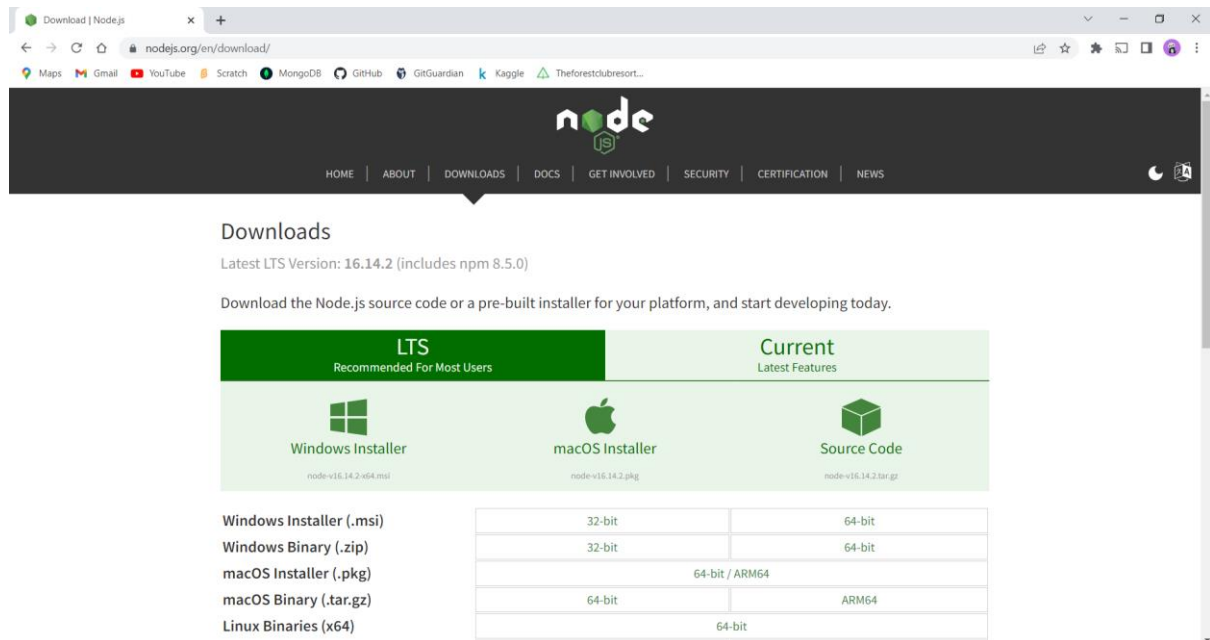
### **Theory:**

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications.

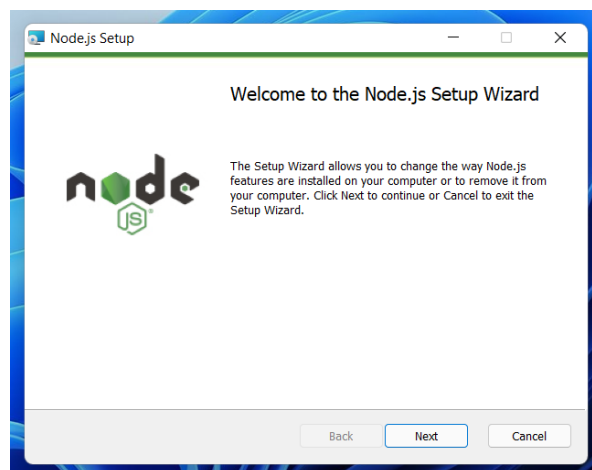
NPM: Node Package Manager, is a package manager for the JavaScript programming language maintained by npm, Inc. NPM is the default package manager for the JavaScript runtime environment Node.js. It is an application and repository for developing and sharing JavaScript code.

### **Step 1: Download Node.js Installer**

In a web browser, navigate to <https://nodejs.org/en/download/>. Click the **Windows Installer** button to download the latest default version. At the time this article was written, version 10.16.0-x64 was the latest version. The Node.js installer includes the NPM package manager.



## Step 2: Install Node.js and NPM from Browser



1. Once the installer finishes downloading, launch it. Open the **downloads** link in your browser and click the file. Or browse to the location where you have saved the file and double-click it to launch.
2. The system will ask if you want to run the software – click **Run**.
3. You will be welcomed to the Node.js Setup Wizard – click **Next**.
4. On the next screen, review the license agreement. Click **Next** if you agree to the terms and install the software.
5. The installer will prompt you for the installation location. Leave the default location unless you have a specific need to install it somewhere else – then click **Next**.


6. The wizard will let you select components to include or remove from the installation. Again, unless you have a specific need, accept the defaults by clicking **Next**.

7. Finally, click the **Install** button to run the installer. When it finishes, click **Finish**.

### **Step 3: Verify Installation**

Open a command prompt and enter the following commands: **node -v**

The system should display the Node.js version installed on your system. You can do the same for NPM: **npm -v**



```
Command Prompt
Microsoft Windows [Version 10.0.22000.556]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lekha khobrekar>node -v
v14.17.2

C:\Users\lekha khobrekar>npm -v
6.14.13
```

**Conclusion:** We have Downloaded and installed Node.js

### **Questions:-**

1. What is nodejs and NPM

## **Experiment No: 5**

**Aim:** To create first application using Node.js

**Lab Outcome:** Students will able to develop first node js program.

**Theory:**        **Node.js**

- It is an open source server environment.
- Node.js runs on various platforms (Windows, Linux, etc.)
- It uses JavaScript on the server.

### **Why Node.js**

- Node.js uses asynchronous programming.

A common take for a web browser can be used to open a file or a web browser and return the content to the client.

Here is how node.js handles a file request.

- Sends the task to the computer file system.
- Ready to handle the next request.

When the file system has opened and read the file servers, it returns the content to the client. Node.js eliminates the waiting and simply continues with the request.

- Node.js runs single-threaded asynchronous programming which is very memory efficient. Node.js is file
- A Node.js contains tasks that will be executed on certain events.
- Node.js file must be initiated on the server before having effect.
- Node.js file has extension ‘.js’

### **Source Code:**

Program to write ‘Hello World’ in Node.js

```
var http = require('http');
http.createServer(function(req,res)
{
res.writeHead(200,{‘Content-Type’:”text/html”});
res.end(‘Hello World’);
})
listen(8080)

var http = require('http');
http.createServer(function(req,res)
```

```
{  
res.writeHead(200,{‘Content-Type’:”text/html”});  
Console.log(“This example is different”);  
Console.log(“The results is displayed on command line”);  
res.end();  
})  
listen(8080);
```

**Conclusion:** From this experiment, we have learnt the needs of Node.js, its uses, file system and how to write hello world program in Node.js

**Questions:**

1. Features of Node.js
2. Why we use Node.js

## **Experiment No. 6**

**Aim:** Creating and using custom components and using the styled components library for CSS to React

**Lab Outcome:** Students will able to use custom components and styled components react Js.

### **Theory:**

Styled components is the result of wondering how we could enhance CSS for styling React component system. By focusing on a single on a single use case we managed to optimize the experience for developers as well as the output for end users.

Apart from the improved experience for developers styled-components provides:

- Automatic critical CSS
  - No class name bugs
  - Easier deletion of CSS
  - Simple dynamic styling
  - Painless maintenance
  - Automatic vendor prefixing
- Installation
- Installing styled-components only takes a single
- npm install –save styled-components
- Styled-components utilizes tagged template literals to style your components. It removes the mapping between components and styles. This means that when you're defining your styles, you're actually creating a normal. React components, that has your styles attached to it.
- Source Code:
- Step 1: Create a react app using the following command on the command prompt.
- ```
.npx create-react-app styled-components
```
- Step 2: After creating the app:

- Cd styled-components  
➔ Change directory
- npm start  
➔ this will start your app

Step 3: Write the following example code

App.js

```
import './App.css';  
import styledButton from './components/Button';  
function App()  
{  
  return(  
    <div className="App">  
      <styledButton> Styled Button </styledButton>  
    </div>  
  );  
}  
export default App;
```

Button.js

```
import styled from 'styled-components';  
const styledButton = styled.button`  
border: 2px solid #4caf50;  
background-color: #4caf50;  
color: white;  
padding: 15px 32px;  
text-align: center;  
display: inline-block;  
font-size: 16px;  
cursor: pointer;  
transition: 0.55s all ease-out;
```



```
export default styleButton;
```

Save the src/App.js file to get the output.

**Conclusion:** In this experiment, we learnt about styled components in react application ie they are CSS in JavaScript tools that bridges the gap between components and styling and also implemented the code for the same.

**Questions:-**

1. Explain styled components of react.js
2. Explain Custom components of react.js

## **Experiment No.7**

**Aim:** To use React to Hooks: useState and use Ref.

**Lab outcome:** Student will able to organize side effects in a component using react hooks

### **Theory:**

#### - React Hooks

Hooks are the new feature introduced in the React 16.8 version. It allows you to use state and other React features without writing a class. Hooks are the functions which “hook into” React state and Lifecycle features from function components. It does not work inside classes. Hooks are backward-compatible, which means it does not contain any breaking changes. Also it does not replace your knowledge of React concepts.

#### – When to use Hooks

If you write a function component, and then you want to add some state to it, previously you do this by converting it to a class. But now you can do it by using Hook inside the existing function component.

#### – useState Hook:

- The react use-State Hook allows us to track state in a function component. State generally refers to data or properties that need to be tracked in an application.

- To use a useState Hook, we first need to import it into our component.

Ex: import {useState} from “react”;

- We initialize our state by calling useState in our function component. useState accepts an initial and returns two values.

- The current state.

- A function that updates the state.

Ex: import {useState} from “react”;

Function FavouriteColor();

Const[color, setColor] = useState();

}

Notice that again, we are destructuring the returned values from useState. The first value, color, is our current state. The second value, setColor is the function that is used to update our state. Lastly, we set the initial to an empty string: useState (“”);

### Procedure:

Step 1: Create a react app using the following command on the command prompt.

\* npx create-react-app use-state

Step 2: After creating the App with the following command: useState

– change directory

npm start

– this will start your app

Step 3: Write the following code.

Source Code:

App.js

```
import {useState} from "react";
```

```
import ReactDOM from "react-dom/client";
```

```
function FavCity()
```

```
{
```

```
  const [city, SetCity] = useState({  
    continent: "Europe";  
    countries: "France";  
    city: "Paris";
```

```
});
```

```
const updateCity = () => {
```

```
  setCity(previousState => {  
    return { ...previousState, City: "Cannes" }
```

```
  });
```

```
  return (
```

```
    <>
```

```
    <h1>{city.continent}</h1>
```

```
    <p>
```

```
    Favorite City {city.city} in Favorite Country {city.countries}
```

```
    </p>
```

```
    <button type="button"
```

```
    onClick = {updateCity}>
```

```
    Cannes </button>
```

```
  );
```

```
}
```

```
const root = ReactDOM.createRoot(document.getElementById("root"));
```

```
root.render(<FavCity/>);
```

```
export default FavCity;
```

– useRef Hook:

The useRef Hook allows you to persist values between renders. It can be used to store mutable values that do not cause a re-render when updated. It can be used to access a DOM element directly.

- If we tried to count how many times our application renders using the useState Hook, we would be caught in an infinite loop since this Hook itself causes a re-render. To avoid this, we can use the useRef Hook.
- useRef() only returns one item. It returns an object caused by current. When we initialize useRef we set the initial value: useRef(0);

Procedure:

Step 1: Create a react app using the following command on the command prompt.

```
* npx create-react-app use-ref
```

Step 2: After creating the App run the following commands.

- cd use-ref
  - change directory
- npm start
  - this will start your app.

Step 3: Write the following example code.

Source Code:

App.js

```
import {useState,useEffect,useRef} from "react";
import ReactDOM from "react-dom/client";
function App()
{
    const[inputValue,outputValue] = useState(" ");
    const count = useRef(0);
    useEffect(). => {
        count.current = count.current + 1;
    });
    return (
        <>
        <input type="text" value={inputValue} onChange = {(e)} => {
            setInputValue(e.target.value)} />
        <h1>Render Count:{count.current} </h1>
    )
}
```

```
};  
}
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(<App/>);  
export default App;
```

**Conclusion:** From this experiment, we have learnt React Hooks and how Hooks make React so much better because we have simpler code that implements similar functionalities faster and more effectively. We also saw example programs to implement useState and useRef hooks and have a better understanding of it.

**Questions:-**

1. Explain React hooks
2. Explain why we use React hooks

## **Experiment No. 8**

**Aim:** Create a basic to do app in React Js using concepts as needed.

### **Theory:**

In React, hooks are functions that allows you to hook into React state and lifecycle features from function components. This allows you to use React without classes.

- `useState`: returns a stateful values.
- `useEffect`: perform side effects from functions components.
- `useContext`: accepts a context objects and returns current context value.
- `useCallback`: pass an inline callback and an array of dependencies.

The only hook we will need for this particular todo list project is `useState()`. This hook replaces the need for a state object in a class component structure.

Our goal is to create a todo list UI. This UI will have three main components:

- Header that labels the Todo List UI. This is just basic application identifier.
- A list to display each to do item.
- A form that adds a todo task to the list. The default complete should be set to false.

### **Procedure:**

1. Create a React application

`npm create-react-app todo-list`

`cd` into the todo-list and then `npm start`.

Your project should be now started on `localhost:3000`

2. App.js

Navigate to App.js and get rid of everything between the two `<div>` tags. We wont need any of the pre populated code. Our App.js is pretty base bones at this point.

```
import React from "react";
```

```
import './App.css'
```

```
function App(){
```

```
  return (
```

```
    <div className='App'>
```

```
      Hello! Todo List will go here!
```

```

        </div>
    );
}
export default App;

```

### 3. Header:

Create a file in the src directory and name it Header.js. Then create a presentational component that will display a header identifying the name of your application. Export your header and import it to App.js. In the empty <div>, add <Header />

```

import React from "react";
import './App.css'
const Header = () => {
    return (
        <header> <h1> Todo List </h1> </header>
    );
}
export default Header;

```

### 4. Create mock data to list application

Create a data.json file in the src directory in App.js add import data from “./data.json”; to your list of imports at top of page.

```

data.json
{
    "id": 1;
    "task": "Giving dog a bath";
    "complete": true;
}

```

Create a few tasks with incrementing ids and complete attributes as true or false.

### 5. Read list of dos and display:

The next thing we need to do is test out ability to read a set of test data. Use out useState() hook to wire up some local site in App.js

Syntax: const[variable, setVariable] = useState ({initState});

Now we need to map over the todoList and create individual to called Todolist.js and Todo.js

The Todolist.js file is the container that holds all of our todos and Todo.js is one single row in Todolist.

Export the Todolist and import to App.js. Also export the Todo and import it to Todolist.js

```
import React, { useState } from 'react';
import data from './data.json';
import Header from './Header';
import Todolist from './Todolist';
import './App.css';
function App() {
  const [todoList, setTodoList] = useState(data);
  return (
    <div className = 'App'>
      <Header />
      <Todolist todoList = { todoList } />
    </div>
  );
}
```

export default App;

Because our state logic is held in App.js we need to pass our entire todoList down to our <Todolist /> component.

Todolist.js

```
import React from 'react';
import Todo from './Todo';

const Todolist = ({ todoList }) => {
  const [todoList, setTodoList] = useState(data);
  return (
    <div className = 'App'>
      {todoList.map(todo => {
        return (
          <Todo todo = { todo } />
        );
      })}
    </div>
  );
}
```

export default Todolist;

Todo.js

```
import React from 'react';
const Todo = ({ todo }) => {
```



```

        return (
          <div>
            {todo.task}
          </div>
        );
      }
    }
    export default Todo;

```

#### 6. Toggle task completion:

Let's tackle toggling on and off whether or not a task is completed.

Let's first add a className to our individual Todolist component that will help us with styling

```

const Todo = ({todo}) => {
  return (
    <div className = {todo.complete?"strike":""}>
      {todo.task}
    </div>
  );
}

```

We will use a className strike to enforce styling in our index.css, add

```

.strike{
  text-decoration: line-through;
}

```

In App.js create a toggle function(toggle()) is fairly simple.

```

const handleToggle = (id) => {
  return (
    let mapped = todoList.map(task => {
      return task.id == id? {... task, complete: !
task.complete}: {...task};
    });
    setTodolist(mapped);
  );
}

```

#### 7. Deleted complete task

What are we going to do with all of these crossed off completed task?

Let's delete them. Create a button that will have an onClick handler that filters all the completed items.

```

const handleFilter = () => {
  return (
    let filtered = todoList.filter(task => {
      return ! task.complete;
    });
  );
}

```

```

    });
    setTodolist(filtered);
  );
}

```

Add a button to the end of the Todolist component and set onClick to the handleFilter. Add your handler filter function to App.js and then pass down the function as props to the Todolist.

Todolist.js

```

import React from 'react';
import Todo from './Todo';
const TodoList = ({todoList, handleToggle, handleFilter} => {
  return (
    <div> {todoList.map(todo => {
      return (
        <Todo todo = { todo } handleToggle = {
handleToggle } handleFilter = { handleFilter } />
      );
    })}
    <button stlye = {{margin: '20px'}} onClick =
{handleFilter} >
      Clear completed
    </button>
  </div>
  );
});
export default Todolist;

```

#### 8. Add tasks with form component:

The final item on our list to create a form component that will handle adding tasks to our Todolist. Create a new file in your src directory and call it to yourTodoForm.js

```

import React, { useState } from 'react';
const TodoForm = ({addTask}) => {
  const[userInput, setUserInput] = useState("");
  const handleChange = (e) => {
    set userInput(e.currentTarget.value)
  }
  const handleSubmit = (e) => {
    e.preventDefault();
    addTask(userInput);
    setUserInput("");
  }
}

```

```
    }  
    return(  
      <form onSubmit = {handleSubmit}>  
        <input value = {userInput} type = "text" onChange =  
{handleChange} placeholder = "Enter the task...">  
        <button>Submit</button>  
      </form>  
    );  
  }  
  export default TodoForm;
```

**Conclusion:** From this experiment I learnt to make or create a react app i.e. Todo app using react hooks. This application uses majority useState() hook.

**Questions:-**

1. Explain how to Create a basic to do app in React Js

## **Experiment No.9**

**Aim:** Use Github to create a custom build workplace.

### **Theory:**

You only need a Github to create and run a Github action workflow. In this experiment you add a workflow that demonstrates some of the essential features of Github actions.

The following examples show you how Github actions jobs can be automatically triggered, where they run and they can interact with the code in your repository.

Creating your first workflow:

- Create a github / workflow repository directory in your repository on Github. In the Github/workflow directory creates a file named github-actions-demo.xml
- Copy the following yaml contains into the github-actions-demo.yml
- Scroll to the bottom of the page and select. Create a new branch for this commit and start a pull request. Then to create a pull request, click props new file committing the workflow file to branch in your repository triggers the push event and runs your workflow.

Viewing your workflow results:

- On github.com, navigate to the main page of the repository.
- Under your repository name click actions
- In the left sidebar, click the workflow you want to see.
- From the list of workflow runs, click the name of the run you want to see.
- Under jobs, click the explore Github-Actions job.
- The log shows you how each of the steps was processed and expand any of the steps to view i.e. details.

**Conclusion:** From this experiment performed how to create the first workflow in Github and how to view the workflow results.

### **Questions:-**

1. Explain how to create your own Work flow
2. Explain how to view your Work flow results

## **Experiment No: 10**

**Aim:** To install and configure Splunk, upload sample data using Splunk and execute search command in Splunk web.

### **Theory:**

Splunk is an innovative technology which searches and indexes log files and helps organization derive insights from the data. A main benefit of Splunk is that it uses indexes to store data, and so does not require a separate store for its information. Splunk is used for monitoring and searching through big data it indexes and correlates information in a container that makes it searchable and its possible to generate alerts, reports and visualization. IF can recognize data patterns, create metrics and helps diagnose problems for business challenges like IT Management, Security and Compliance.

### **Installation of Splunk**

Step 1: Navigate to the folder or directly where the installer is located.

Step 2: Double click the Splunk MSI file to start the installer.

Step 3: In the welcome panel, read the license agreement and click on Check this box to accept the license agreement and click on Next.

Step 4: A terminal window appears and you are prompted to specify on administrator user id and password to use with the splunk trial.

Step 5: Click Next.

Step 6: (Optional) You are prompted to create a shortcut on the start menu. If you want to do this. Click Create.

Start Menu Shortcut.

Step 7: Click Install.

Step 8: In the installation complete panel, confirm that the launch browser with splunk checkbox is selected.

Step 9: Click Finish, the installation finishes.

Step 10: Splunk enterprise starts and Splunk web launches in a browser window.

Step 11: Go to the steps to launch Splunk Web.

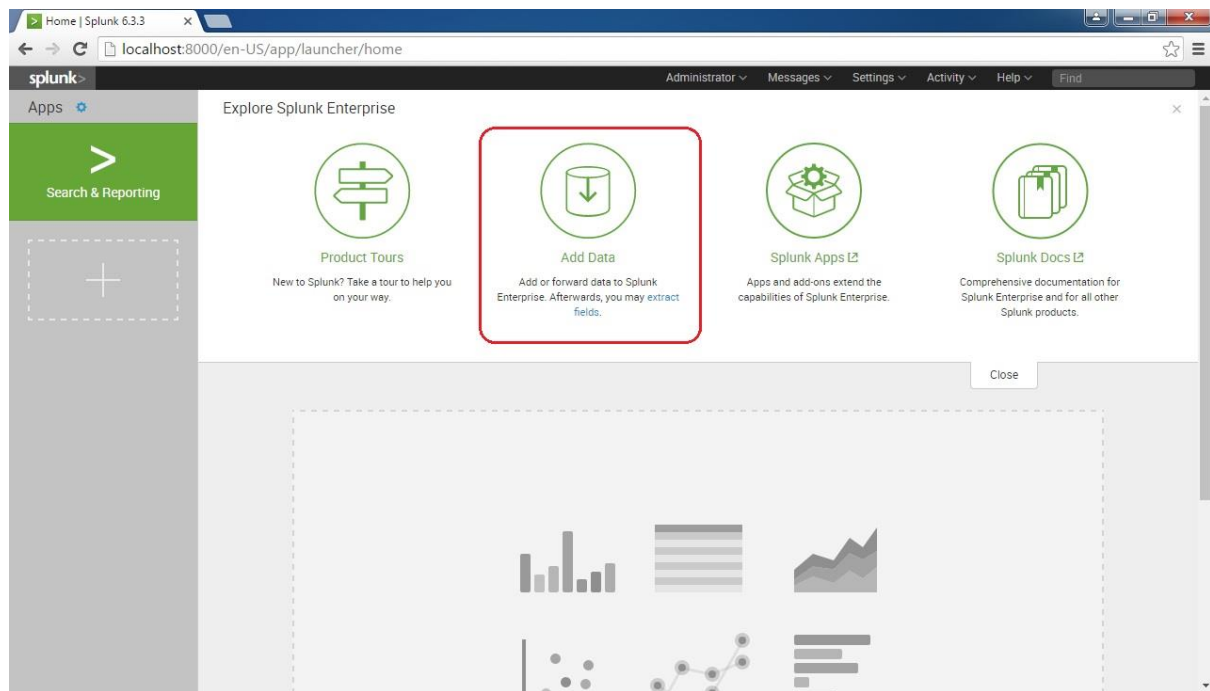
The screenshot displays the Splunk documentation page for the 'Splunk Dashboards app (beta) for Enterprise and Cloud'. The page includes a navigation bar with links to Products, Solutions, Why Splunk?, Resources, and Splunkicon. The main content area is titled 'Install the Splunk Dashboards app (Beta)' and provides a list of prerequisites and steps for installation. The steps are as follows:

- Download Splunk Dashboards app (beta) from Splunkbase.
- Log into your Splunk platform instance.
- In the apps panel, click the gear icon.
- Click **Install app from file**.
- Select the Splunk Dashboards app file.
- Confirm that you want to restart the Splunk platform instance to complete the installation, if needed. The installation does not require restart, however.

Additional steps include clicking **Done** when the installation completes and clicking **Next** to proceed to the 'Set source type' page.

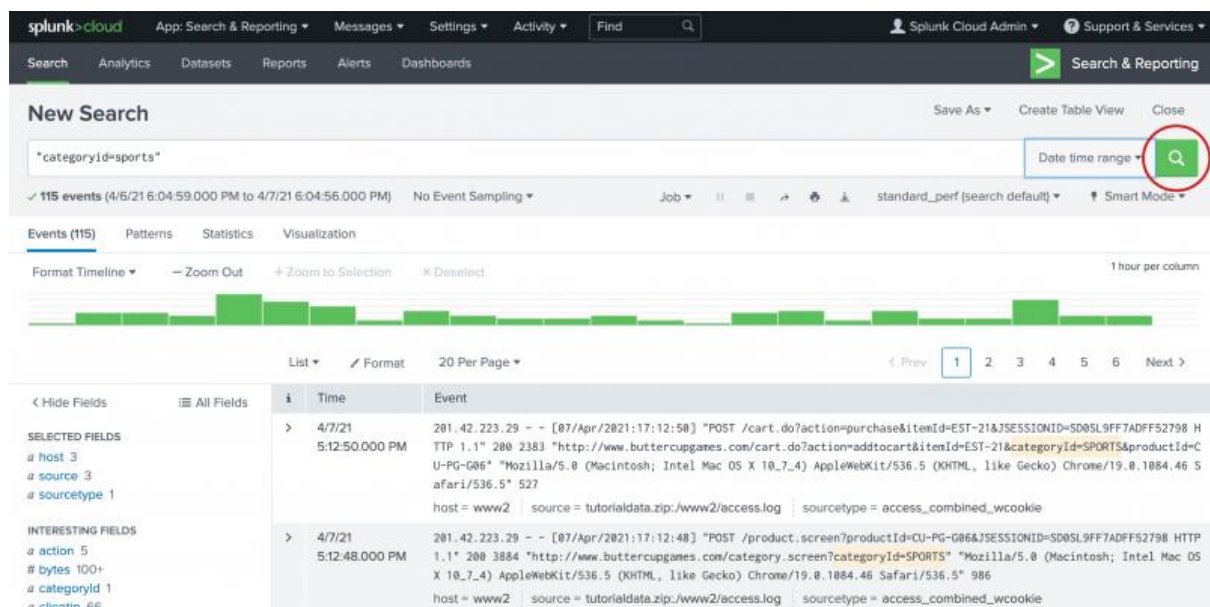
## Upload data using Splunk

1. To create a text index where you can store text data. Click Settings > Indexes.
2. Click new index and assign the index a name. To minimize resources consumption, specify a small size and retention period.
3. Select settings > Add Data.
4. Click Upload.
5. Click Select File, browse to a log file on your computer and click Open. The file is uploaded, Click Next.
6. On the set source type page, select the correct source type for the file your uploaded or if none is appropriate, specify a name for the new sources type and click Next.
7. On the input settings page, select your test index.
8. Click Review and verify your settings.
9. Click Submit.



## Search Command

1. Click search in the App bar to start a new search.
2. Type buttercup in the search bar.
3. When you type a few letters into the search bar, The Search Assistant shows you terms in your data that match the letters that you type in.
4. From the list of workflow runs, Click the name of run you want to see.
5. Under jobs, click the explore – GitHub – Actions job.
6. The log shows you how each of the steps was processed. Expand any of the steps to view i.e. details.



**Conclusion:** From this experiment, we learnt how to create first workflow in GitHub and how to view the workflow results.

**Questions:-**

1. Explain Splunk
2. Explain the steps to installation of Splunk