

Practical No.1: Setup environment for Angular framework by Installing Node.js, npm package manager using editor like Visual Code.

A. Objective: Setting up environment for first time execution for angular practicals is important task before we start our angular practical's programming. It consist of many installations like visual code editor, Node.js and npm package manager. Once it is installed we are ready to use our setup environment to make project in angular.

B. Expected Program Outcomes (POs)

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the digital electronics engineering problems.
2. **Design/ development of solutions:** Design solutions for digital electronics engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
3. **Engineering Tools, Experimentation and Testing:** Apply modern digital electronics engineering tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

1. Install visual code open source software.
2. Setup environment for angular practical execution using node js and npm package manager.

D. Expected Course Outcomes(Cos)

Prepare environment for angular project using Node.js, npm and visual code editor.

E. Practical Outcome(PRo)

Setup environment for angular practical execution with NODE JS and NPM Package manager.

F. Expected Affective domain Outcome(ADos)

1. Follow Coding standards and practices.
2. Maintain tools and equipment.
3. Follow safety practices.
4. Follow ethical practices

G. Prerequisite Theory:

Angular is basically is an open-source, JavaScript-based client-side framework that helps us to develop a web-based application. Actually, Angular is one of the best frameworks for developing any Single Page Application or SPA Applications.

Angular is a UI framework for building mobile and desktop web applications. It is built using javascript framework for front-end development. It uses Typescript language (Superset of javascript) to make angular application. You can build amazing client-side applications using HTML, CSS, and Typescript using Angular. It is maintained by Google and a community of experts acting as a solution for rapid front-end development.

Learner can have following roles after learning Angular in a company.

- Web developer
- Web app developer
- UI developer
- UX developer
- Front-end developer
- JavaScript developer

Now, let's start process of Environment setup to install angular framework. Following tools/packages are required to run angular project.

- Nodejs
- Npm
- Angular CLI
- IDE for writing your code

Here are the steps needs to be followed to Environment setup in angular.

Step1: Install Node.js and npm:

Node.js is tool to run the development server for an Angular application. This allows developers to make changes to the code and see the updates in real-time without having to manually reload. It can be used as a build tool to automate tasks such as compiling TypeScript to JavaScript, bundling the application code, and optimizing the code for production.

Node.js can also be used to create a backend API that an Angular application can consume. Overall, Node.js can greatly enhance the development and deployment process for Angular applications.

Npm stands for Node Package Manager, which is a package manager for the Node.js runtime environment. npm is used to manage packages and dependencies for Node.js applications, including Angular applications. In an Angular application, npm is used to install and manage packages that the application depends on, such as Angular itself, third-party libraries, and development tools. These packages are typically stored in the "node_modules" directory of the application.

npm provides a command-line interface that allows developers to install, update, and remove packages, as well as manage package versions and dependencies.

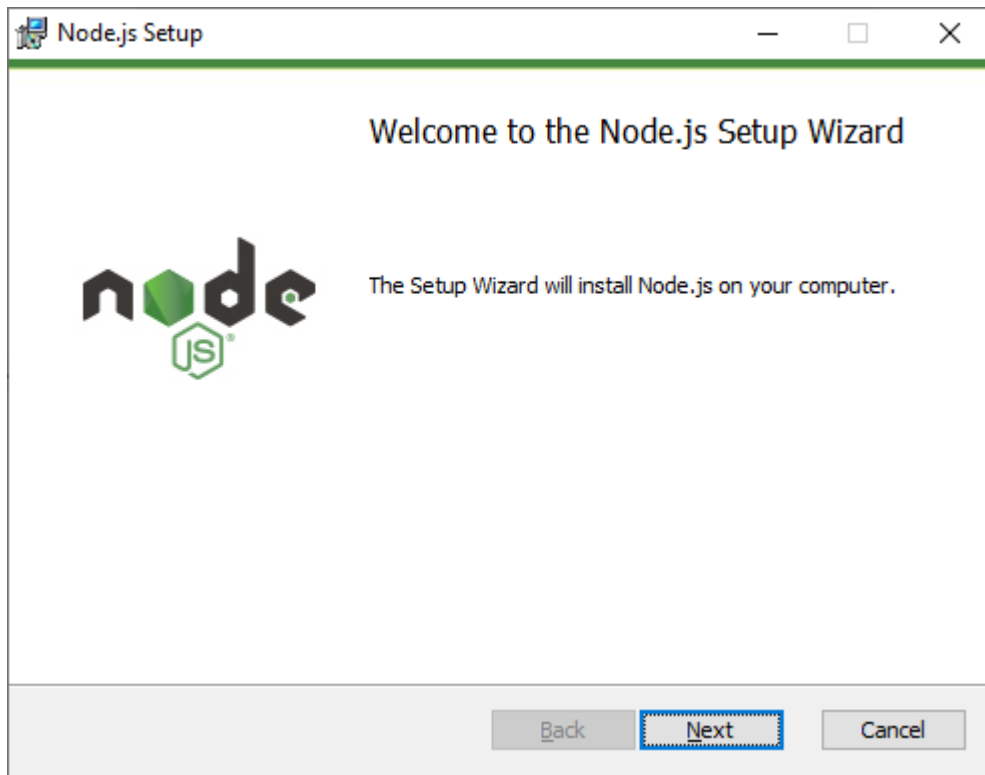
Overall, npm plays a critical role in the development and management of Angular applications, making it easier for developers to build and maintain their applications and collaborate with others in the community.

To install Node.js, Go to <https://nodejs.org/en/> and download the latest version of Node.js that corresponds to your operating system.

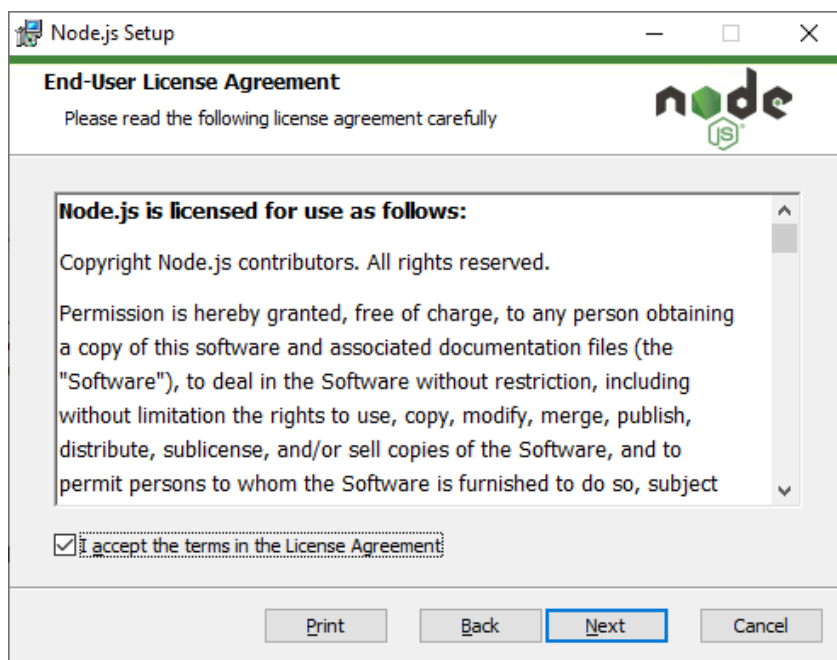
The screenshot shows the Node.js download page. The header includes the Node.js logo and navigation links: HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. The main content area is titled "Downloads" and features a "Latest LTS Version: 18.13.0 (includes npm 8.19.3)" and a "Current Latest Features" section. Below this, there are three main download options: "Windows Installer" (node-v18.13.0-x64.msi), "macOS Installer" (node-v18.13.0.pkg), and "Source Code" (node-v18.13.0.tar.gz). A table lists additional platforms: 32-bit, 64-bit, 64-bit / ARM64, ARM64, ARMv7, and ARMv8. The table also includes a link to the source code: node-v18.13.0.tar.gz.

LTS Recommended For Most Users		Current Latest Features
Windows Installer node-v18.13.0-x64.msi	macOS Installer node-v18.13.0.pkg	Source Code node-v18.13.0.tar.gz
Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v18.13.0.tar.gz	

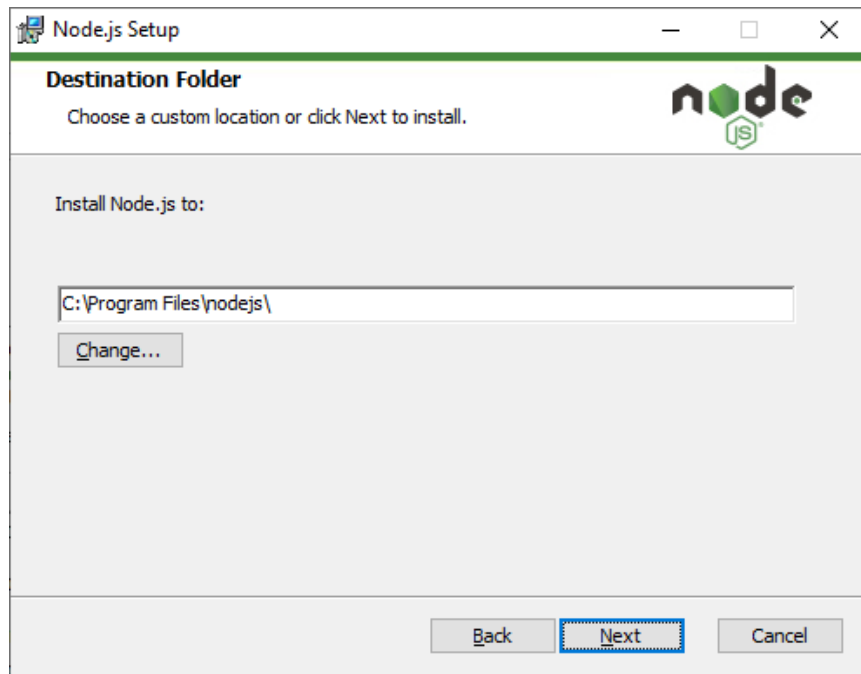
After download nodejs, Run the downloaded installer file and Follow the installation wizard to install Node.js on your computer.



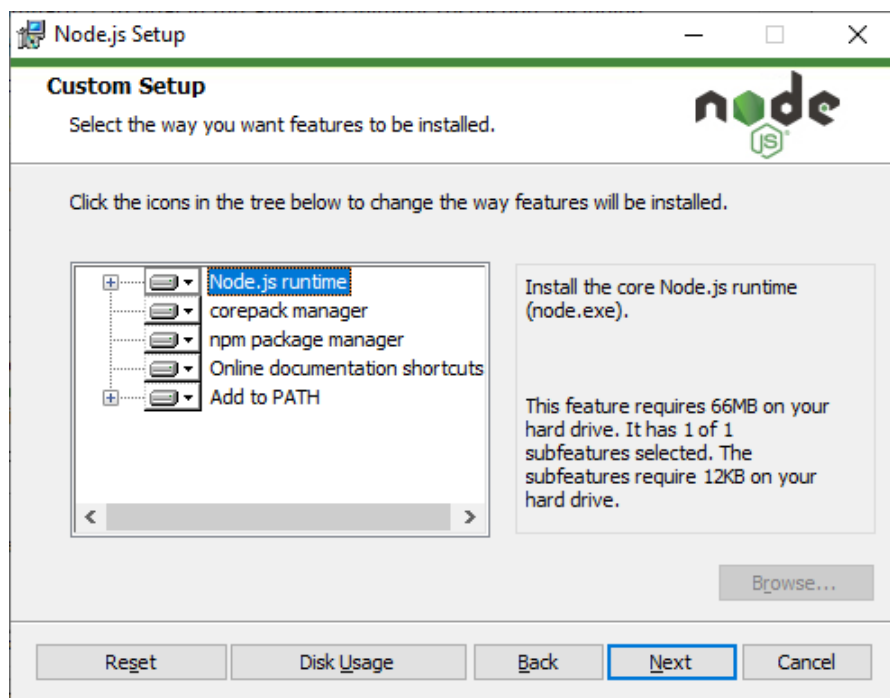
When click on next button, it will ask to accept End-User Licence Agreement as shown below.



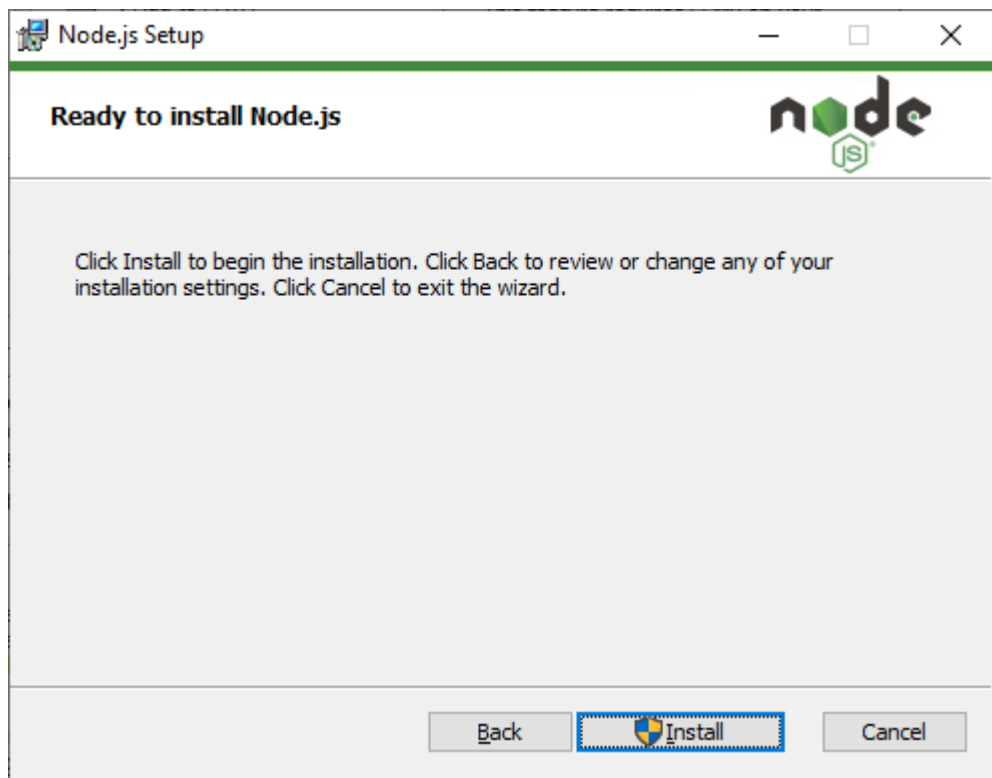
Once you check in checkbox and click on Next Button, you will get prompt box for node.js installation directory. Here you can see default location path as shown in below however you can customize your location by click on change button option.



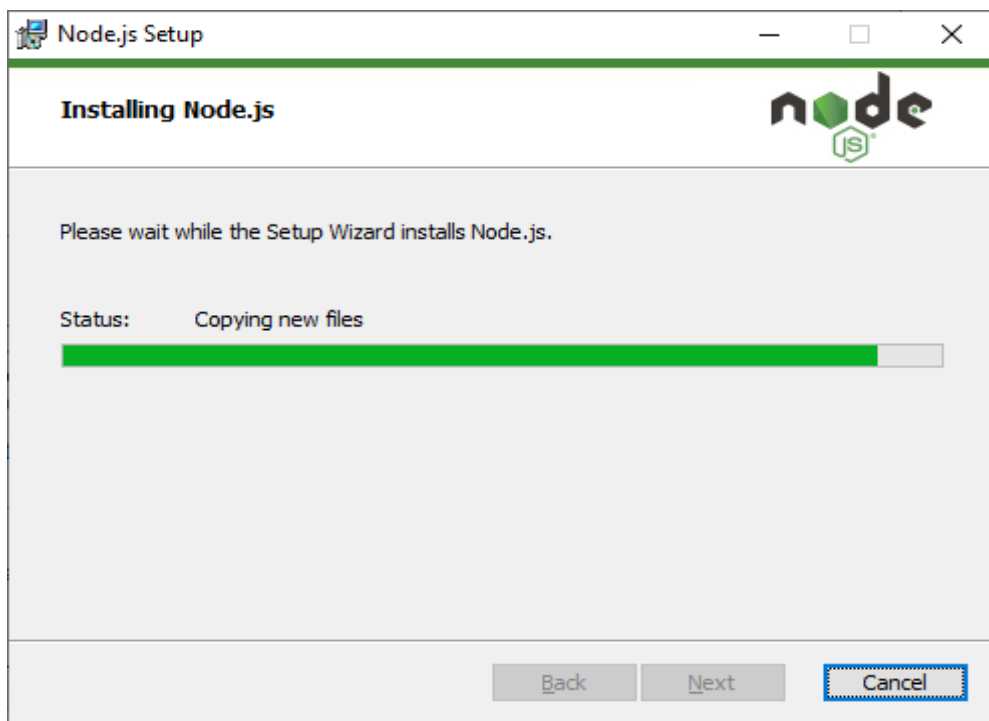
After choose directory, you can select features that you want to customize as shown in below.



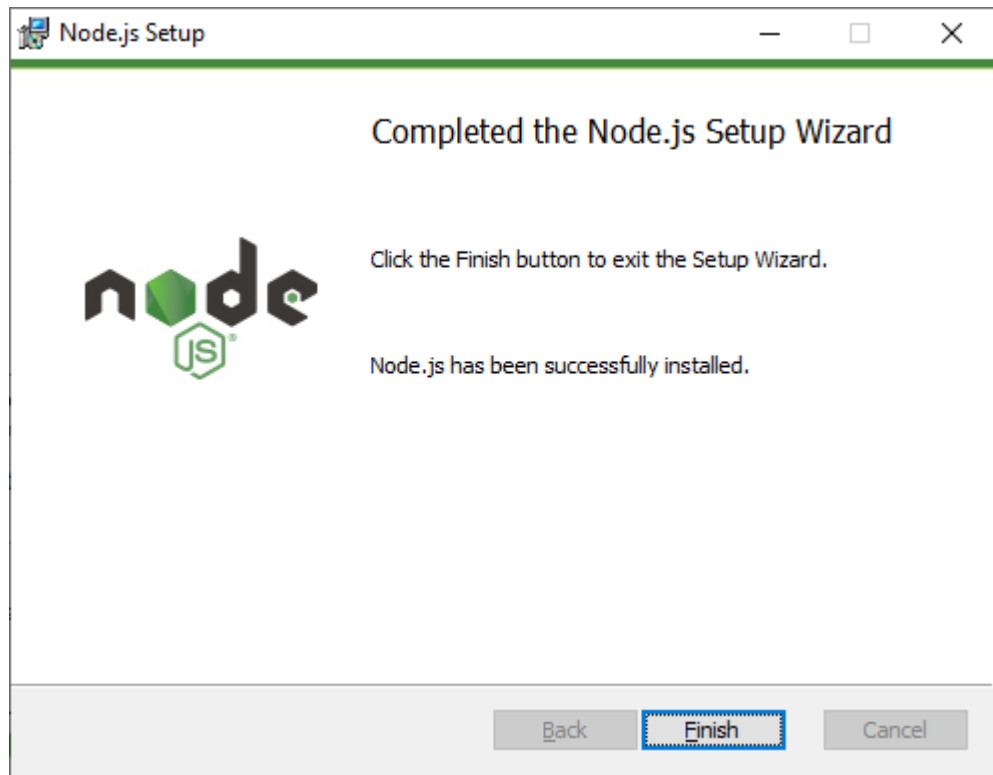
Here npm package manager is also installed automatically as you can see from above figure. After select custom features and click on next button, Node.js Setup wizard ready to install node.js and npm as shown below.



When you click on Install button, wizard will start for installation as shown in below. Wait for the installation to complete. This may take a few minutes.



Once the installation is complete, click "Finish" to close the installer as shown below.

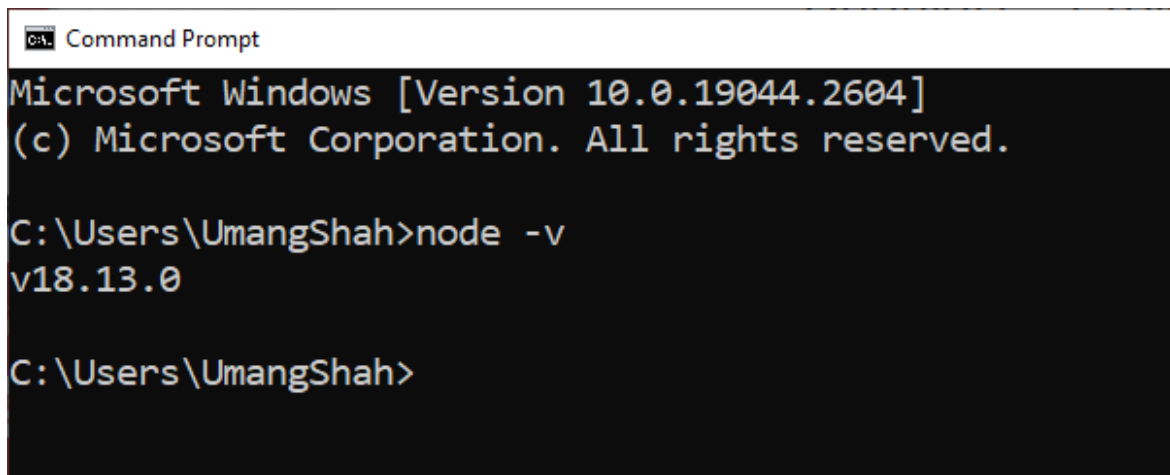


On successful installation, the command prompt will be displayed like this:

```
C:\Windows\system32\cmd.exe - node
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\UmangShah>node
Welcome to Node.js v18.13.0.
Type ".help" for more information.
>
```

You can Verify that Node.js has been installed correctly by opening a command prompt and typing "node -v" This should display the version of Node.js that you have installed as shown below.

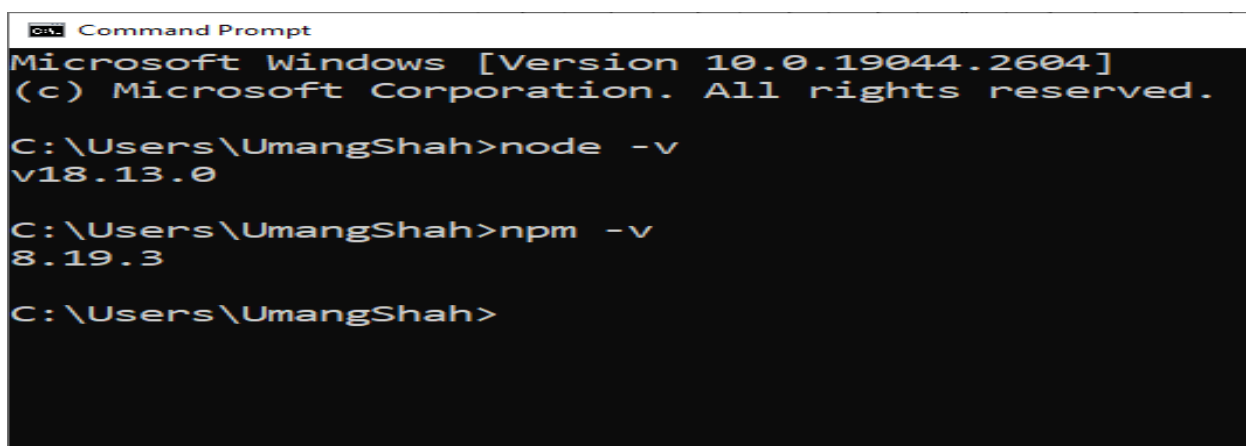
A screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt". The window content shows the Windows version and copyright information: "Microsoft Windows [Version 10.0.19044.2604] (c) Microsoft Corporation. All rights reserved." The user is at the prompt "C:\Users\UmangShah>" and has entered the command "node -v". The output is "v18.13.0".

```
Command Prompt
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\UmangShah>node -v
v18.13.0

C:\Users\UmangShah>
```

You can verify that npm (Node Package Manager) has been installed correctly by opening a command prompt and typing "npm -v". This should display the version of npm that you have installed as shown below.

A screenshot of a Windows Command Prompt window, similar to the one above. It shows the same Windows version and copyright information. The user is at the prompt "C:\Users\UmangShah>" and has entered the command "npm -v". The output is "8.19.3".

```
Command Prompt
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\UmangShah>node -v
v18.13.0

C:\Users\UmangShah>npm -v
8.19.3

C:\Users\UmangShah>
```

Step2: Install Angular CLI:

Angular CLI is used to create, build, and manage your Angular projects.

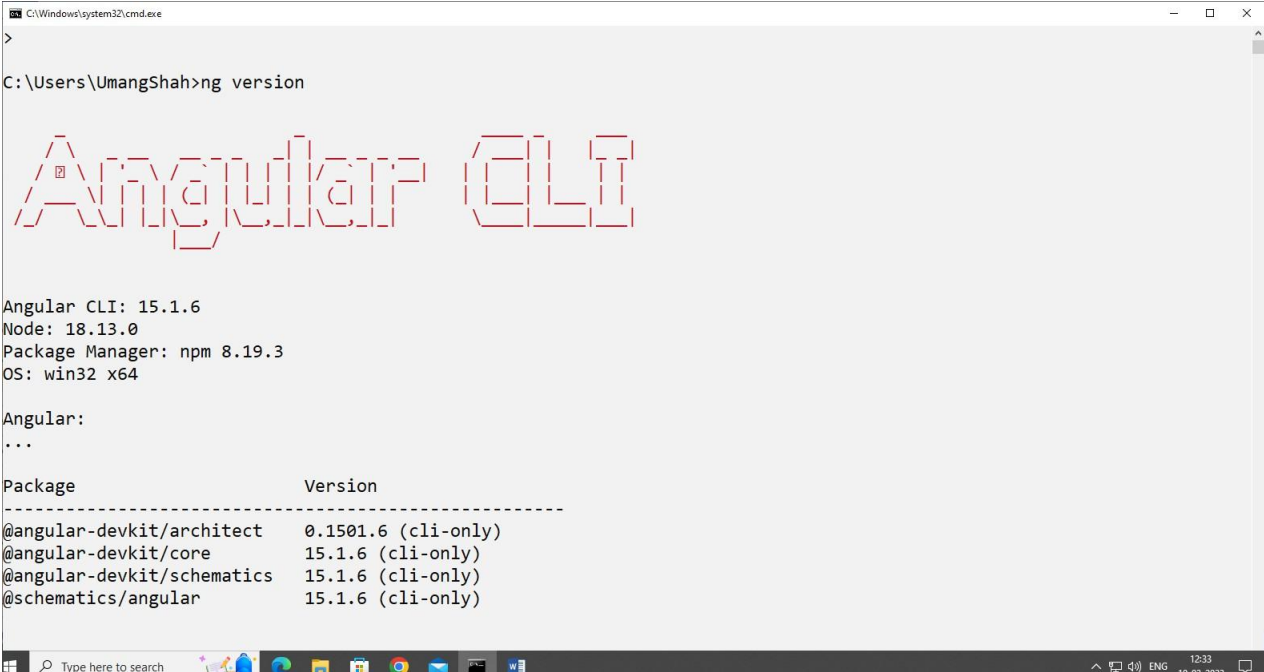
Once Node.js is installed, open a command prompt or terminal window and run the following command to install the Angular CLI

```
npm install -g @angular/cli
```


Wait for the installation to complete. This may take a few minutes depending on your internet speed. Once the installation is complete, run the following command to verify that the Angular CLI has been installed correctly.

`ng version`

You should see the version number of the Angular CLI displayed in the console output as given below.



```
C:\Windows\system32\cmd.exe
>
C:\Users\UmangShah>ng version

Angular CLI
Angular CLI: 15.1.6
Node: 18.13.0
Package Manager: npm 8.19.3
OS: win32 x64

Angular:
...

Package      Version
-----
@angular-devkit/architect 0.1501.6 (cli-only)
@angular-devkit/core      15.1.6 (cli-only)
@angular-devkit/schematics 15.1.6 (cli-only)
@schematics/angular       15.1.6 (cli-only)
```

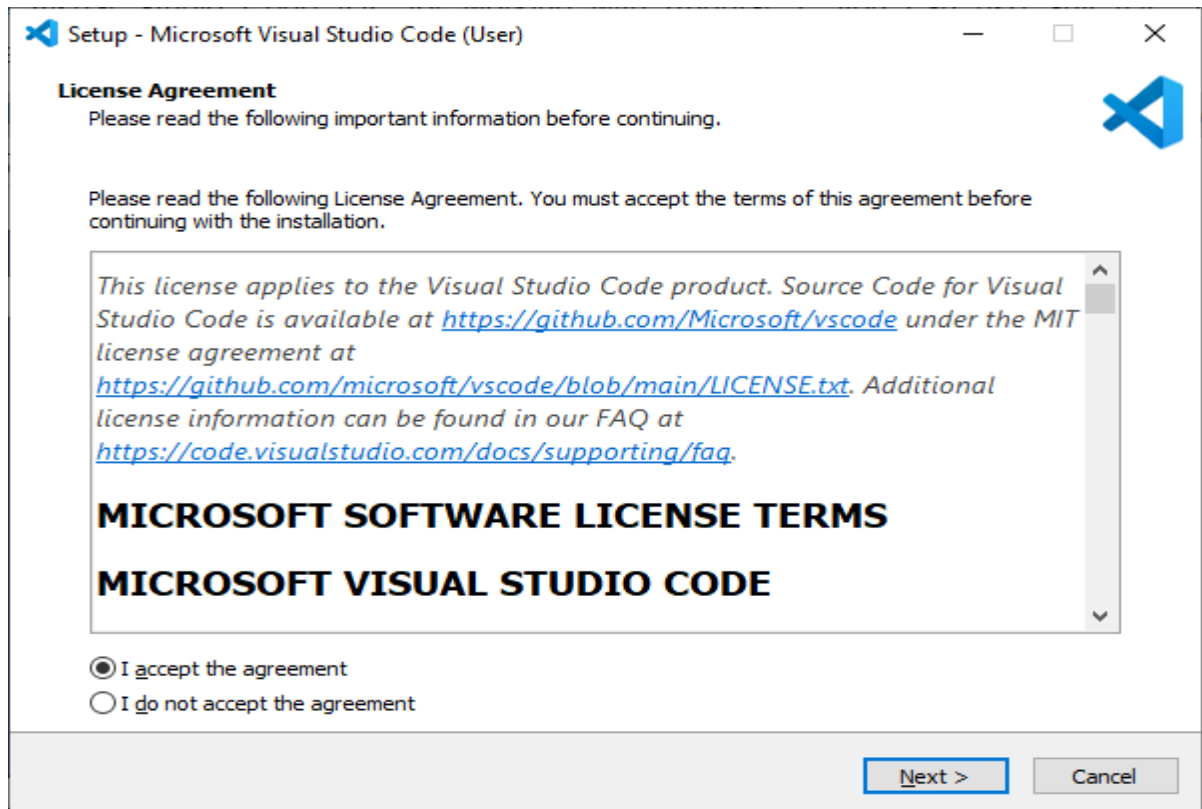
You have successfully installed the Angular CLI on your system. It gives the version for Angular CLI, typescript version and other packages available for Angular.

We are done with the installation of Angular.

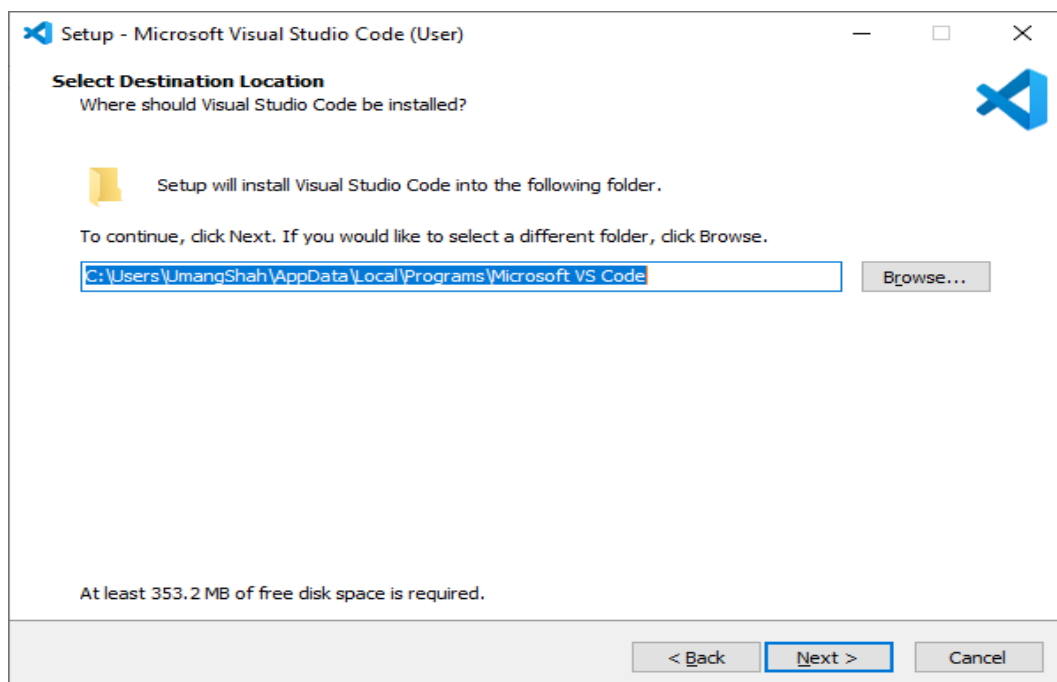
Step3: Install Visual Studio Code Editor:

Following steps are needed to perform for installing Visual Studio Code.

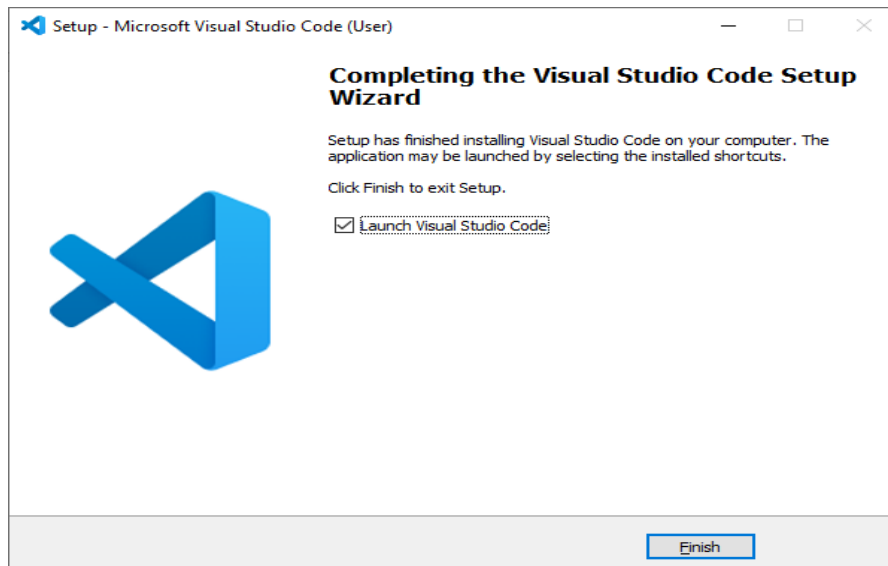
1. Go to the official Visual Studio Code website at <https://code.visualstudio.com/>
2. Click on the "Download for Windows" button to download the Windows version of Visual Studio Code.



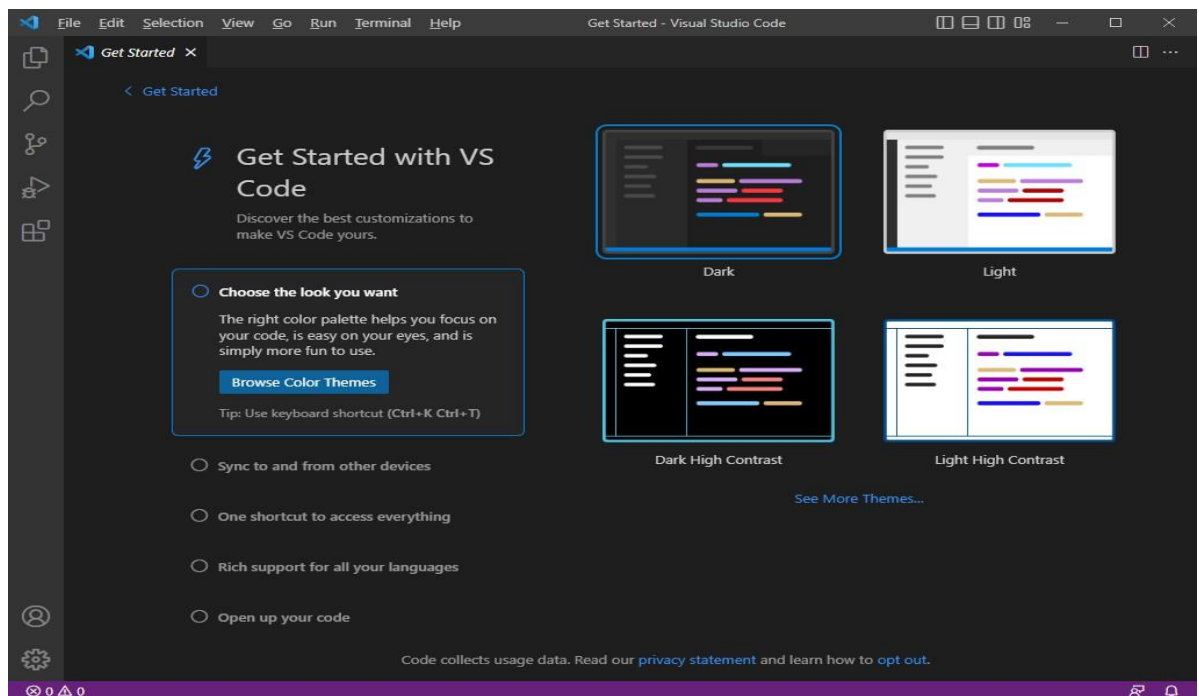
3. Run the downloaded installer.
4. Click on the "Next" button to begin the installation process.
5. Choose the installation location and click on the "Next" button.



6. Choose the start menu folder and click on the "Next" button.
7. Choose the additional tasks you want the installer to perform and click on the "Next" button.
8. Click on the "Install" button to start the installation process.
9. Wait for the installation process to complete.



10. Once the installation is complete, click on the "Finish" button to exit the installer.
11. Launch Visual Studio Code by double-clicking on the desktop icon or by searching for "Visual Studio Code" in the Windows Start menu.



You have successfully installed Visual Studio Code on your Windows computer for manage Angular application.

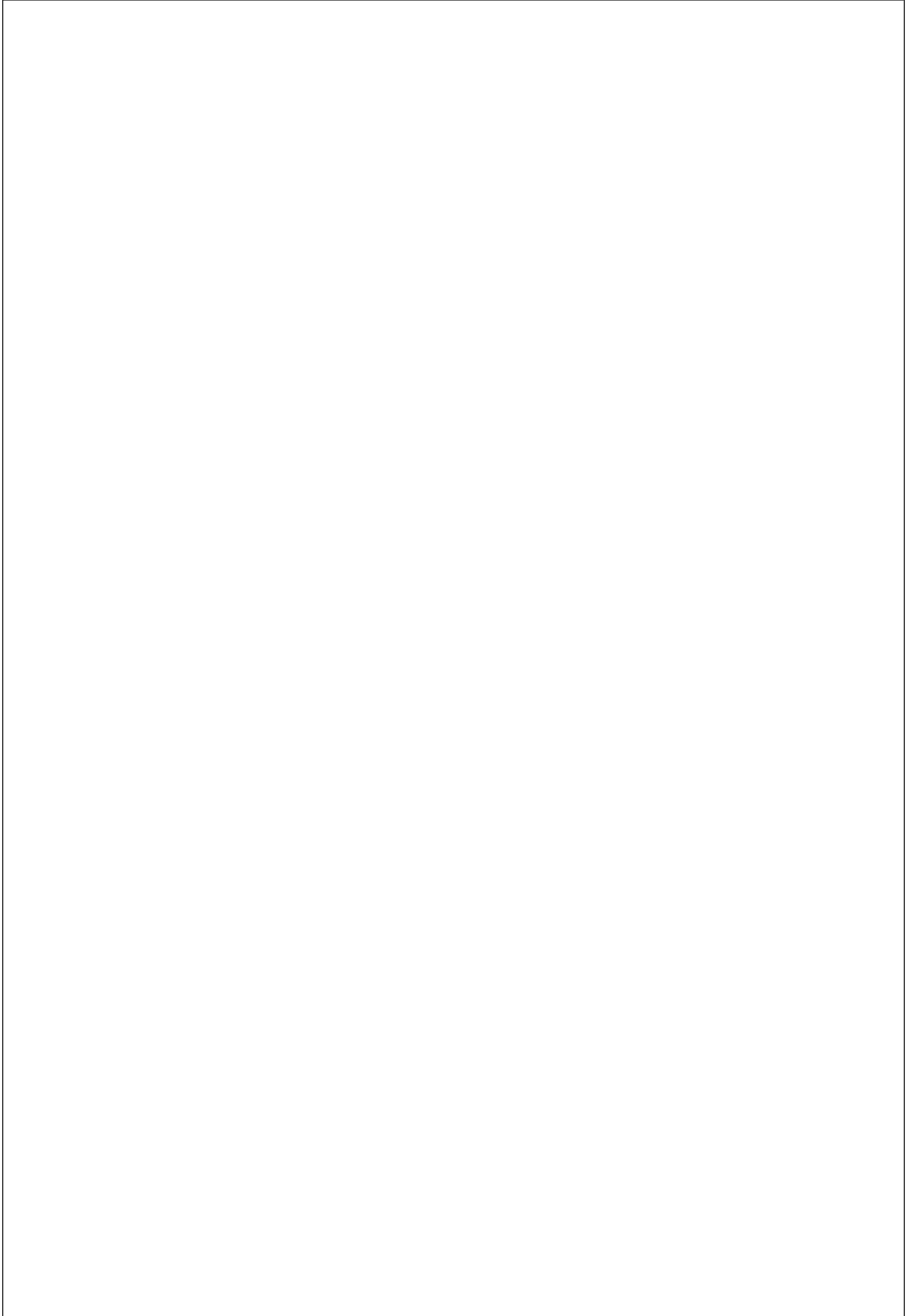
H. Resources/Equipment Required

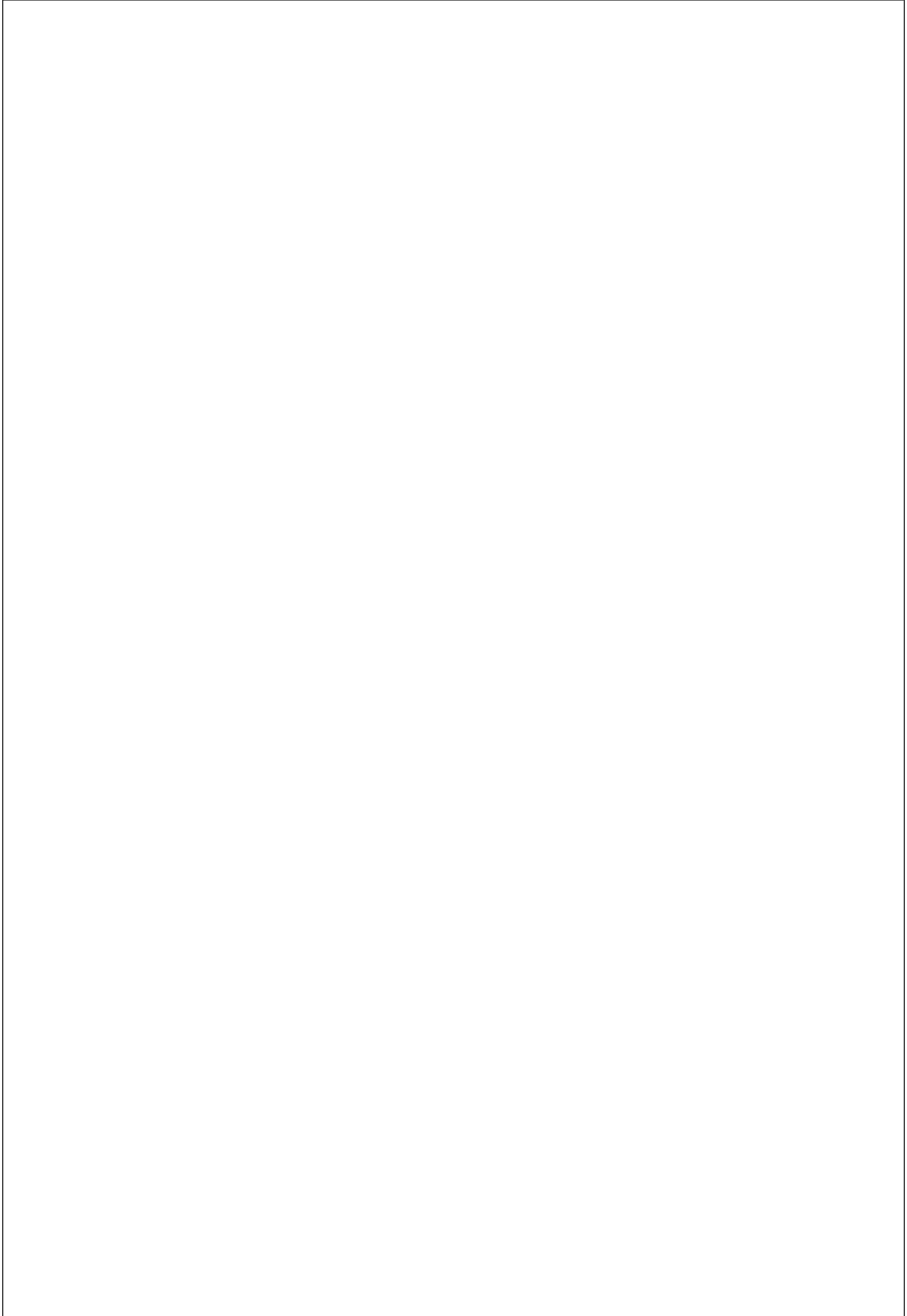
S r. N o.	Equipment/ Software Resources	Specification
1	Computer System	Intel I3 processor with minimum 4 GB RAM, 40GB HDD, Windows 7 or above Operating system.
2	Visual Code	Open source software from Microsoft
3	Node JS and NPM Package Manager	Open source software
4	Browser	Microsoft Edge, Google Chrome etc

I. Output Source code:

J. Practical related Questions.

1. Why Angular is more popular than other framework?
2. Describe NPM in detail.
3. How do you verify that Node.js is installed correctly on your system?
4. How do you verify that the Angular CLI is installed correctly on your system?





Practical No.2: Create first application to print Hello World message using angular framework.

A. Objective:

The objective of this program is a simple that is starting point for learning a new programming language or framework that is to display a simple message or output on the screen to demonstrate the basic functionality of the framework.

B. Expected Program Outcomes (POs)

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the digital electronics engineering problems.
2. **Design/ development of solutions:** Design solutions for digital electronics engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
3. **Engineering Tools, Experimentation and Testing:** Apply modern digital electronics engineering tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

1. Project Environment setup for run application in Angular framework.
2. Display Data on web application using Angular framework.

D. Expected Course Outcomes(Cos)

Setup environment for angular practical execution with NODE JS and NPM Package manager.

E. Practical Outcome(PRo)

Create first application to print Hello World message using angular framework.

F. Expected Affective domain Outcome(ADos)

1. Follow Coding standards and practices.
2. Maintain tools and equipment.
3. Follow safety practices.
4. Follow ethical practices

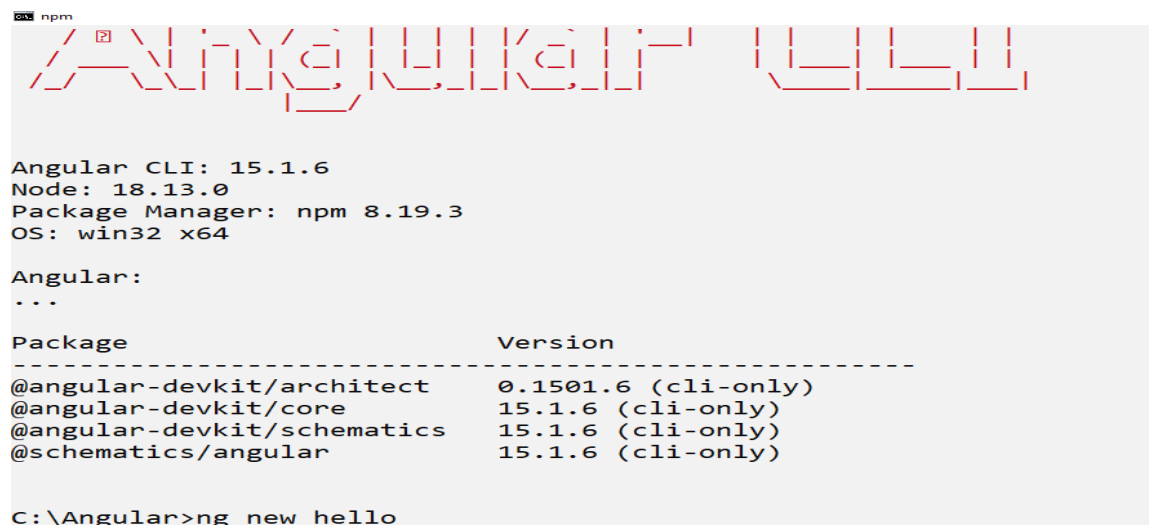
G. Prerequisite Theory:

To create a project in Angular After setup environment, go to directory where you already installed Angular/Cli and type following commands using either terminal of visual studio or command line.

ng new projectname

Here, You can use the projectname of your choice.

Let us now run the above command in the command line.



```


Angular CLI: 15.1.6
Node: 18.13.0
Package Manager: npm 8.19.3
OS: win32 x64

Angular:
...

Package                Version
-----
@angular-devkit/architect 0.1501.6 (cli-only)
@angular-devkit/core      15.1.6 (cli-only)
@angular-devkit/schematics 15.1.6 (cli-only)
@schematics/angular       15.1.6 (cli-only)

C:\Angular>ng new hello
  
```

Once you run the command it will ask you about routing as shown below –



```

Angular CLI: 15.1.6
Node: 18.13.0
Package Manager: npm 8.19.3
OS: win32 x64

Angular:
...

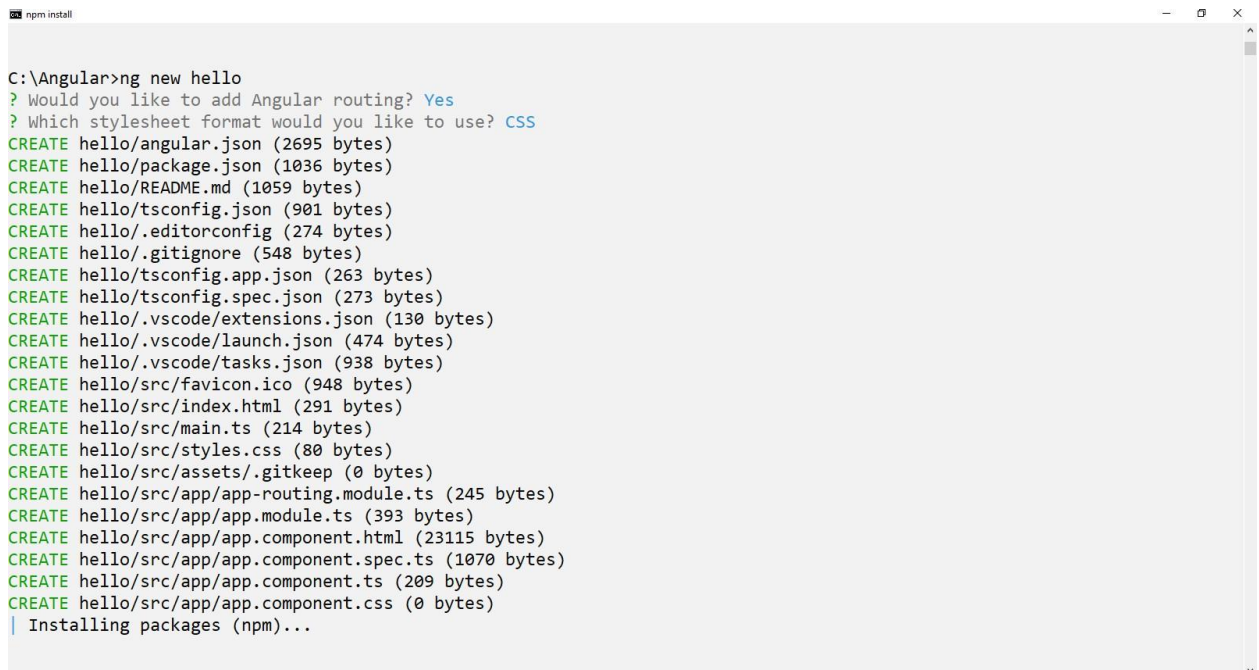
Package                Version
-----
@angular-devkit/architect 0.1501.6 (cli-only)
@angular-devkit/core      15.1.6 (cli-only)
@angular-devkit/schematics 15.1.6 (cli-only)
@schematics/angular       15.1.6 (cli-only)

C:\Angular>ng new hello
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use?
> CSS
SCSS [ https://sass-lang.com/documentation/syntax#scss ]
Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
Less [ http://lesscss.org ]
  
```

Type y to add routing to your project setup and again ask question for the stylesheet as shown below.

The options available are CSS, Sass, Less and Stylus. In the above screenshot, the arrow is on CSS. To change, you can use arrow keys to select the one required for your project setup. At present, we select CSS option for our project-setup.

It installs all the required packages necessary for our project to run in Angular as shown below.

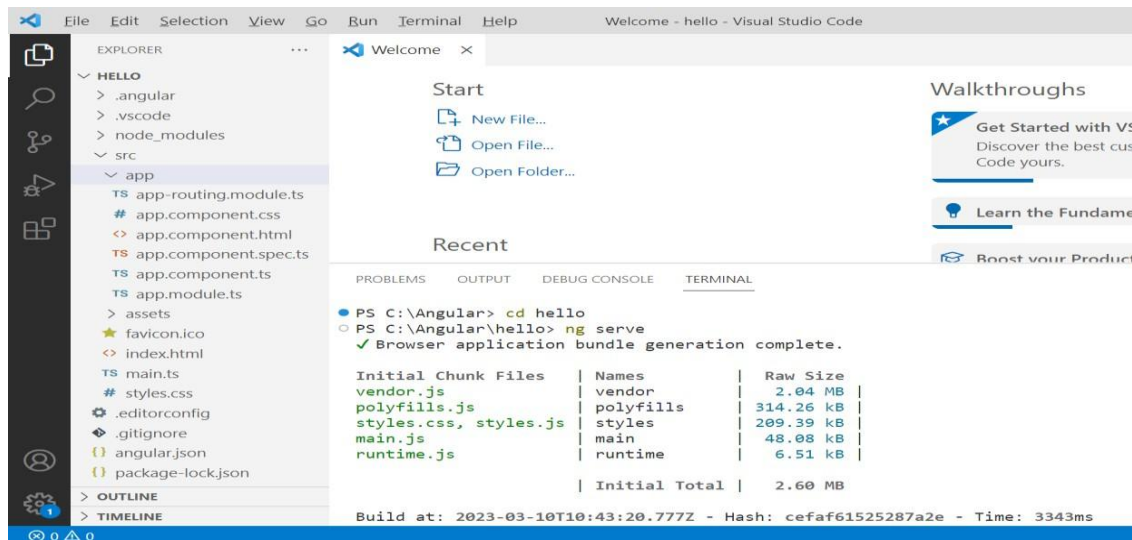


```
npm install
C:\Angular>ng new hello
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE hello/angular.json (2695 bytes)
CREATE hello/package.json (1036 bytes)
CREATE hello/README.md (1059 bytes)
CREATE hello/tsconfig.json (901 bytes)
CREATE hello/.editorconfig (274 bytes)
CREATE hello/.gitignore (548 bytes)
CREATE hello/tsconfig.app.json (263 bytes)
CREATE hello/tsconfig.spec.json (273 bytes)
CREATE hello/.vscode/extensions.json (130 bytes)
CREATE hello/.vscode/launch.json (474 bytes)
CREATE hello/.vscode/tasks.json (938 bytes)
CREATE hello/src/favicon.ico (948 bytes)
CREATE hello/src/index.html (291 bytes)
CREATE hello/src/main.ts (214 bytes)
CREATE hello/src/styles.css (80 bytes)
CREATE hello/src/assets/.gitkeep (0 bytes)
CREATE hello/src/app/app-routing.module.ts (245 bytes)
CREATE hello/src/app/app.module.ts (393 bytes)
CREATE hello/src/app/app.component.html (23115 bytes)
CREATE hello/src/app/app.component.spec.ts (1070 bytes)
CREATE hello/src/app/app.component.ts (209 bytes)
CREATE hello/src/app/app.component.css (0 bytes)
| Installing packages (npm)...
```

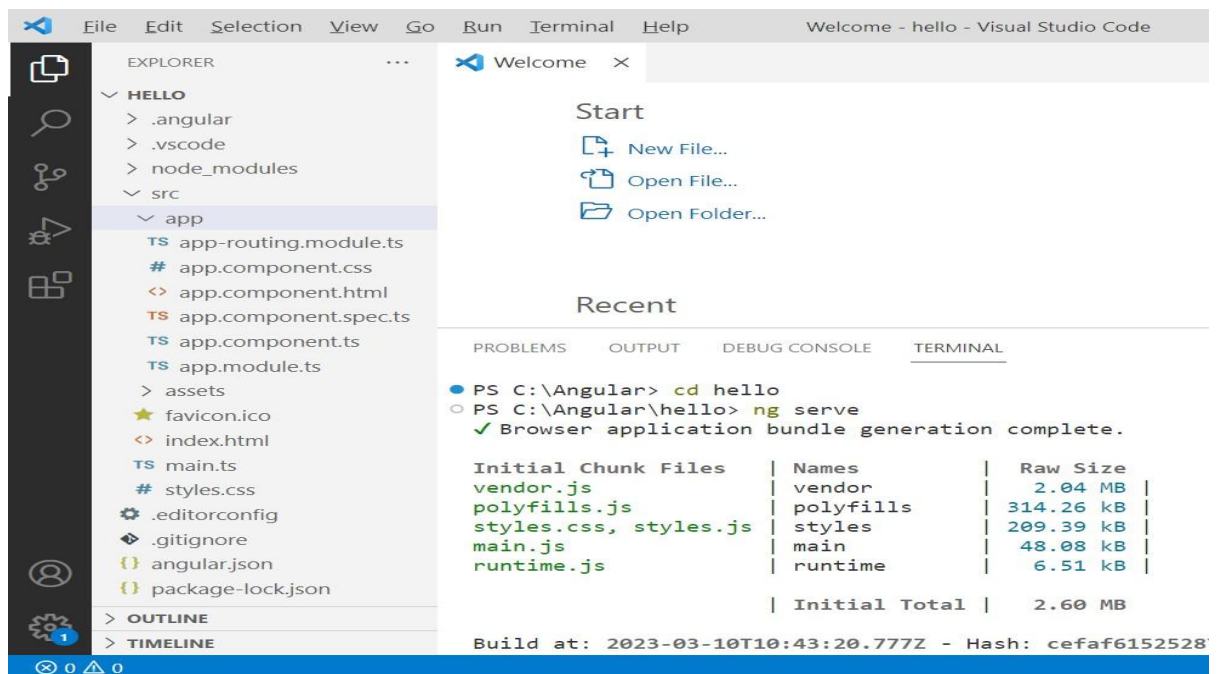
Once all packages installed, Our project is installed successfully. We need to go our project directory using following command

cd projectname

Now I am going to show you using Visual code editor. You can see above command in our example as given below.



If you want to see the file structure of our project where you can manage your application, open Visual studio code editor which is already installed and open our project directory, you will get the folder structure that looks like the one given below.



To compile our project, the following command is used in angular.

ng serve

The ng serve command builds the application and starts the web server.

You will see the below output when the command starts executing.

The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left showing a project structure. The main editor area displays the output of the terminal. The terminal shows the following commands and output:

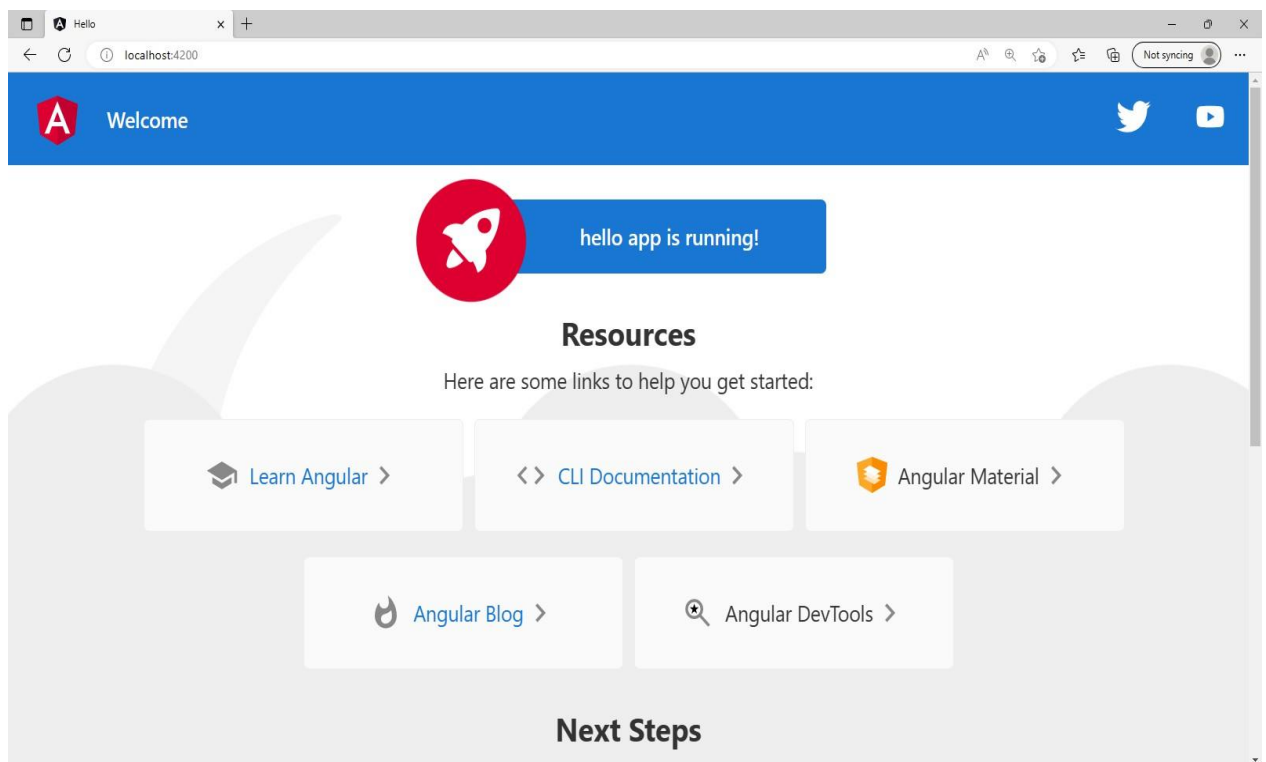
```
PS C:\Angular\hello> cd..
PS C:\Angular> cd hello
PS C:\Angular\hello> ng serve
✓ Browser application bundle generation complete.
```

Initial Chunk Files	Names	Raw Size
vendor.js	vendor	2.04 MB
polyfills.js	polyfills	314.26 kB
styles.css, styles.js	styles	209.39 kB
main.js	main	48.08 kB
runtime.js	runtime	6.51 kB
Initial Total		2.60 MB

Build at: 2023-03-10T10:43:20.777Z - Hash: cefaf61525287a2e - Time: 3343ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

Once the project is compiled and type url, <http://localhost:4200/> in the browser, you will see your project home page as shown below.



The necessary files for this app are given below:

1. **app.module.ts**: This is the root module of an Angular application. It is responsible for importing and configuring all the components, services, directives, and other modules used in the application. The module is typically defined in a file named **app.module.ts**.

Here is an example:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

2. **app.component.ts**: This is the root component of an Angular application. It defines the structure and behavior of the main view of the application. The component is typically defined in a file named **app.component.ts**.

Here is an example:

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

3. **app.component.html**: This is the HTML template for the root component. It defines the layout and content of the main view of the application. The template is typically defined in a file named **app.component.html**. Here is an example:

```
<h1>Hello World!</h1>
```

4. **main.ts:** This is the main entry point of an Angular application. It bootstraps the root module of the application and starts the Angular runtime. The file is typically named main.ts.

Here is an example:

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app.module';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

5. **angular.json:** This is the configuration file for an Angular application. It defines the settings and options for building, serving, and testing the application. The file is typically named angular.json

Now we are going to understand concepts of string interpolation to make our desired output.

String interpolation is a feature in Angular that allows you to embed dynamic values from your component's TypeScript code into your HTML template. This can make your HTML templates more dynamic and interactive. Here's an example of how to use string interpolation in both the HTML and TypeScript files of an Angular component:

HTML Template:

```
<h1>Welcome {{name}}!</h1>
```

TypeScript File:

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

```
@Component({
  selector: 'app-welcome',
  templateUrl: './welcome.component.html',
  styleUrls: ['./welcome.component.css']
})
export class WelcomeComponent {
  name = 'Admin';
}
```

In the HTML template, we've used double curly braces `{{ }}` to surround the name variable, which is defined in the TypeScript file. This tells Angular to replace the `{{name}}` with the value of the name variable when the component is rendered.

In the TypeScript file, we've imported the Component decorator from `@angular/core` and used it to define our component. We've also defined a name variable and set it to 'John'. When the component is rendered, Angular will replace `{{name}}` with the value of name, which is 'Admin'.

You can also use string interpolation to perform simple operations within the template, such as concatenating strings or performing basic arithmetic:

```
<p>{{firstName + ' ' + lastName}} is {{age + 1}} years old</p>
```

Where firstName,lastName and age property variables's value must be declared in TypeScript file.

H. Resources/Equipment Required

S r. N o.	Equipment/ Software Resources	Specification
1	Computer System	Intel I3 processor with minimum 4 GB RAM, 40GB HDD, Windows 7 or above Operating system.
2	Visual Code	Open source software from Microsoft
3	Node JS and NPM Package Manager	Open source software
4	Browser	Microsoft Edge, Google Chrome etc

I. Program Source code Output

Practical No.3: Design a web page to utilize property binding and event binding concepts using button and textbox controls.

A. Objective:

1. To know how to exchange data between the component and the template
2. Create dynamic, interactive web applications that respond to user input and update dynamically based on changes in the component.

B. Expected Program Outcomes (POs)

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the digital electronics engineering problems.
2. **Design/ development of solutions:** Design solutions for digital electronics engineering well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
3. **Engineering Tools, Experimentation and Testing:** Apply modern digital electronics engineering tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

To create dynamic and interactive user interfaces, Manipulating the DOM, Creating responsive UIs, Implementing data-driven applications.

D. Expected Course Outcomes(Cos)

Setup environment for angular practical execution with NODE JS and NPM Package manager.

E. Practical Outcome(Pro)

Design a web page to utilize property binding and event binding concepts using button and textbox controls.

F. Expected Affective domain Outcome(ADos)

1. Follow Coding standards and practices.
2. Maintain tools and equipment.
3. Follow safety practices.
4. Follow ethical practices

G. Prerequisite Theory:

PropertyBinding :

Property **binding** is a feature in Angular that allows you to set the value of an HTML element property to a value from the component. In other words, you can bind a property of an HTML element to a property in your component, and the value of the HTML property will be dynamically updated as the value of the component property changes.

Here is an example of how to use property binding in an Angular component:

HTML Template (app.component.html)

```
<p>Welcome to {{title}}!</p>
<button [disabled]="isDisabled">Click me</button>
```

In this example, we have two different uses of property binding. The first one binds the content of a paragraph (**<p>**) to a property **title** in the component. The curly braces **{{ }}** indicate that this is an example of **one-way data binding**. When the component is rendered, the value of the **title** property is interpolated into the template.

The second use of property binding binds the **disabled** attribute of a button to the **isDisabled** property of the component. Square brackets **[]** are used to indicate that this is an example of property binding. When the value of the **isDisabled** property changes, the **disabled** attribute of the button will be updated accordingly.

TypeScript File (app.component.ts)

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Property binding Demo';
  isDisabled = false;
}
```

In the TypeScript file for our component, we define two properties: **title** and **isDisabled**. The **title** property is a simple string that is used in the first example of property binding to set the content of a paragraph.

The **isDisabled** property is a boolean value that is used in the second example of property binding to disable or enable a button. By default, the **isDisabled** property is set to **false**, so the button is enabled. If we were to change the value of the **isDisabled** property to **true**, the button would become disabled.

By using property binding, we can make our Angular templates more dynamic and interactive. We can easily bind properties of HTML elements to properties in our component, and update them in real time as our data changes.

Event binding:

Event binding is a feature in Angular that allows you to listen for and respond to events that occur in the HTML template, such as button clicks, key presses, or mouse movements. You can bind an event to a method in your component, and when the event is triggered, the method will be called with any relevant data.

Here is an example of how to use event binding in an Angular component:

HTML Template (app.component.html)

```
<button (click)="onClick()">Click me</button>

<p>{{ message }}</p>
```

In this example, we have a button element that is used to trigger an event, and a paragraph element that is used to display a message based on the event. The (click) syntax is used to bind the click event of the button to a method onClick() in the component. When the button is clicked, the onClick() method will be called.

The message that is displayed in the paragraph element is determined by the message property in the component, which is updated by the onClick() method.

TypeScript File (app.component.ts)

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

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

```
export class AppComponent {  
  message: string = "";  
  onClick() {  
    this.message = 'Button clicked!';  
  }  
}
```

In the TypeScript file for our component, we define a property **message** that will be used to display a message in the paragraph element. We also define a method **onClick()** that will be called when the button is clicked.

In the **onClick()** method, we update the value of the **message** property to **'Button clicked!'**. This will cause the message to be updated in the HTML template, and the new message will be displayed to the user.

By using event binding, we can create more interactive and responsive Angular components. We can listen for events in the HTML template and respond to them by updating the properties and methods in our component. This allows us to create dynamic and engaging user interfaces that respond to user input in real time.

ngModel Directive:

ngModel is a built-in directive in Angular that allows you to perform two-way data binding between a form control element and a property in your component. It's typically used with input and select elements to capture user input and update the component's properties with the entered value.

Here's an example of how to use **ngModel** in an Angular component:

HTML Template (app.component.html)

```
<input [(ngModel)]="name" placeholder="Enter your name">  
<p>Your name is: {{name}}</p>
```

In this example, we have an input element that is used to capture the user's name. We use the **[(ngModel)]** syntax to bind the input's value to the **name** property in the component.

This is a two-way data binding, which means that any changes to the input element's value will be reflected in the **name** property, and any changes to the **name** property will be reflected in the input element's value.

We also have a paragraph element that displays the value of the **name** property.

TypeScript File (app.component.ts)

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

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  name: string = '';
}
```

In the TypeScript file for our component, we define a property **name** that will be used to store the user's name. By default, the **name** property is initialized to an empty string.

When the user types a value into the input element, the **name** property will be updated with that value. When the **name** property is updated (for example, if it's updated in code), the input element's value will be updated as well.

By using **ngModel**, we can easily capture user input and update our component's properties in real time. We don't have to manually update the properties or listen for events - **ngModel** takes care of all of that for us. This makes it a very useful tool for creating forms and other interactive user interfaces in Angular.

H. Resources/Equipment Required

S r. N o.	Equipment/ Software Resources	Specification
1	Computer System	Intel I3 processor with minimum 4 GB RAM, 40GB HDD, Windows 7 or above Operating system.

Practical No.4: Create various components of web page using Attribute Directives.

A. Objective:

1. Create a more modular, scalable, and maintainable web page
2. Manipulate and add functionality to HTML elements by binding custom attributes.

B. Expected Program Outcomes (POs)

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 *engineering* 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. **Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological changes *in field of engineering*.

C. Expected Skills to be developed based on competency:

Learner can develop a wide range of technical and soft skills including angular framework knowledge, collaboration etc. making them better equipped to develop high-quality, efficient, and scalable web applications.

D. Expected Course Outcomes(Cos)

Apply angular directives, components and pipes in different web page development.

E. Practical Outcome(PRo)

Create various components of web page using Attribute Directives.

F. Expected Affective domain Outcome(ADos)

1. Follow Coding standards and practices.

2. Maintain tools and equipment.
3. Follow safety practices.
4. Follow ethical practices

G. Prerequisite Theory:

The component is the basic building block of Angular. It has a selector, template, style, and other properties, and it specifies the metadata required to process the component.

Creating a Component in Angular :

To create a component in any angular application, follow the below steps:

- [1] Get to the angular app via your terminal.
- [2] Create a component using the following command:
`ng g c <component_name>`
OR
`ng generate component <component_name>`

Following files will be created after generating the component:

- `component_name.component.html`
- `component_name.component.spec.ts`
- `component_name.component.ts`
- `component_name.component.css`

Now we are going to create a header, sidebar, and footer components in Angular, you can follow these steps:

1. Create a new Angular project:
`ng new my-app`
2. Create a header component using the following command:
`ng generate component header`
3. Create a sidebar component using the following command:
`ng generate component sidebar`
4. Create a footer component using the following command:
`ng generate component footer`

This will create the necessary files for each component, including HTML, CSS, TypeScript, and a spec file as we discussed earlier.

5. Open the `header.component.html` file and add the following code:

```
<header>
<h1>My Website</h1>
</header>
```

6. Open the sidebar.component.html file and add the following code:

```
<aside>
<h2>Menu</h2>
<nav>
<ul>
<li><a href="#">Home</a></li>
<li><a href="#">>About</a></li>
<li><a href="#">Contact</a></li>
</ul>
</nav>
</aside>
```

7. Open the footer.component.html file and add the following code:

```
<footer>
<p>My Website &copy; 2023</p>
</footer>
```

8. Open the app.component.html file and replace the existing code with the following code:

```
<app-header></app-header>
<main>
<app-sidebar></app-sidebar>
<section>
<router-outlet></router-outlet>
</section>
</main>
<app-footer></app-footer>
```

9. Open the app.component.css file and add the following styles:

```
header {
background-color: #333;
color: #fff;
padding: 20px;
}

aside {
background-color: #f2f2f2;
```

```
padding: 10px;
}

footer {
background-color: #333;
color: #fff;
padding: 20px;
text-align: center;
}

main {
display: flex;
}

section {
margin: 20px;
padding: 20px;
border: 1px solid #ccc;
flex-grow: 1;
}
```

You have designed a basic Angular app with a header, sidebar, and footer components.

ngClass directory

ngClass is a built-in Angular directive that allows you to dynamically add or remove CSS classes to an HTML element based on certain conditions. This is particularly useful when you want to apply different styles to an element based on its state or user interaction.

The **ngClass** directive can be used in two ways:

1. Using a string expression:

```
<div [ngClass]="class1 class2 class3">Example</div>
```

In this example, the **ngClass** directive applies three CSS classes to the **div** element: **class1**, **class2**, and **class3**.

2. Using an object expression:

```
<div [ngClass]="{ 'class1': condition1, 'class2': condition2 }">Example</div>
```


In this example, the **ngClass** directive applies the CSS class **class1** to the **div** element if **condition1** is true, and the CSS class **class2** if **condition2** is true. **condition1** and **condition2** value set in ts file.

Here's a more detailed example that demonstrates how to use **ngClass** to dynamically apply CSS classes based on user interaction:

```
<button [ngClass]="{ 'active': isActive }" (click)="toggleActive()">Toggle Active</button>
```

In this example, the **ngClass** directive applies the CSS class **active** to the **button** element if the **isActive** property is true. The **isActive** property is initially set to false in the component:

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

@Component({
  selector: 'app-button',
  template: `
    <button [ngClass]="{ 'active': isActive }" (click)="toggleActive()">Toggle Active</button>
  `
})
export class ButtonComponent {
  isActive = false;

  toggleActive() {
    this.isActive = !this.isActive;
  }
}
```

When the user clicks the button, the **toggleActive()** method is called, which toggles the value of the **isActive** property. When **isActive** is true, the **active** CSS class is added to the button, and when **isActive** is false, the **active** CSS class is removed.

ngStyledirective :

ngStyle is a built-in Angular directive that allows you to dynamically add or remove inline styles to an HTML element based on certain conditions. This is particularly useful when you want to apply different styles to an element based on its state or user interaction.

The **ngStyle** directive can be used in the following way:

```
<div [ngStyle]="{ 'property1': value1, 'property2': value2 }">Example</div>
```

In this example, the **ngStyle** directive applies the CSS styles **property1** and **property2** to the **div** element with their corresponding values **value1** and **value2**.

Here's a more detailed example that demonstrates how to use **ngStyle** to dynamically apply inline styles based on user interaction:

```
<button [ngStyle]="{ 'background-color': isActive ? 'green' : 'red' }"
(click)="toggleActive()">Toggle Active</button>
```

In this example, the **ngStyle** directive applies the inline style **background-color** to the **button** element based on the value of the **isActive** property. If **isActive** is true, the button's background color will be green, and if it is false, the background color will be red. The **isActive** property is initially set to false in the component:

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

@Component({
  selector: 'app-button',
  template: `
    <button [ngStyle]="{ 'background-color': isActive ? 'green' : 'red' }"
    (click)="toggleActive()">Toggle Active</button>
  `,
})
export class ButtonComponent {
  isActive = false;

  toggleActive() {
    this.isActive = !this.isActive;
  }
}
```

When the user clicks the button, the **toggleActive()** method is called, which toggles the value of the **isActive** property. When **isActive** is true, the button's background color will be green, and when **isActive** is false, the background color will be red.

You can also use **ngStyle** in combination with other directives, such as **ngFor** or **ngIf**, to dynamically apply different inline styles to elements based on their state or data.

J. Practical related Exercises.

- I. Design following tourist package webpage using various components of angular or teacher can assign any webpage of website.

