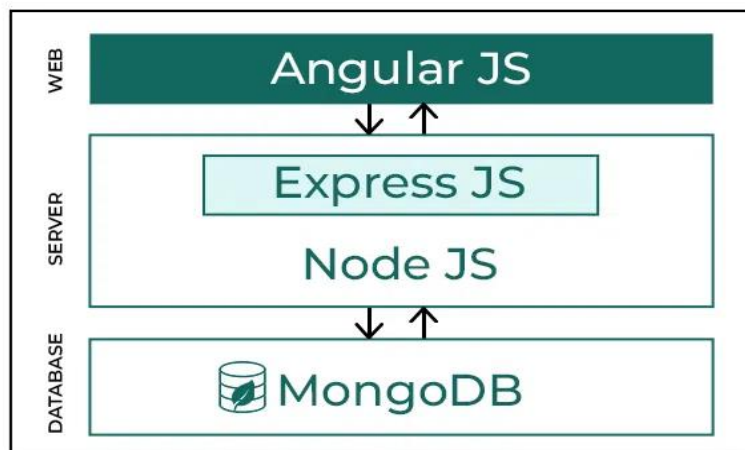


Install MEAN Stack in Windows /Ubuntu and run a simple application.

Installation of Mean in Windows:



How MEAN Stack Works?



MEAN is a software application stack made up of the following components:

- **MongoDB**, a **NoSQL** database with support for server-side **JavaScript** execution
- **Express**, a **Node.js** web application framework
- **Angular**, a web application framework suitable for developing dynamic, single-page applications
- **Node.js**, an asynchronous event-driven framework suitable for building scalable network applications

Valery Karpov invented the term "**MEAN**" and derived the term from the first letter of each element.

Preparing **Windows** to run the **MEAN**

Stack (MongoDB, Express, Angular, Node.js) applications is simple and only requires a couple of things to be installed – namely **Angular-CLI**, **MongoDB** and **Node.js**.

Express runs on top of **Node.js** so it isn't installed directly on **Windows**, it's added via **NPM (Node Package Manager)** when you run `npm install` for an application. The `npm install` command looks at the dependencies section of the `package.json` file for application on the **MEAN stack** and downloads all that is required, which should include **Express**.

Angular is an open-source **JavaScript** front-end web application framework mainly for developing single-page applications. There's some confusion regarding the name and version of **Angular**, initially it was started as **AngularJS** framework (which is still v1), but later it was completely re-written and released as **Angular 2** in September 2016. In March 2017, **Angular 2** was renamed as **Angular**. **Angular** is similar in that it's not installed directly on **Windows** and is added via **NPM**.

Angular-CLI (Command Line Interface), however, will need to be installed globally to create **Angular** projects and makes other development tasks easier. The **Angular CLI** helps us to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

1. Install NodeJS on Windows

~~Download the latest stable release of [NodeJS](#) and install using all the **default options**.~~

Once the installation is complete, verify that the installation was successful by asking **NPM** and **Node** for their version numbers.

```
// Check npm version
npm -version

// This should provide an output similar to
// 6.41

// Check node version
node -v

// This should provide an output similar to
// v10.15.1
```

If you see the version number for both, then the installation was completed successfully.

Follow these steps to install the [Node.js](#) on your Windows:

Step 1: Download Node.js Installer

Visit Our Dept Lab Web Server to download Node for Windows OS

<http://192.168.1.200/node-v22.13.0-x64.msi>

- Visit the [official Node.js](#) website to download the Node.js '.msi' installer *Download NodeJS*

- ~~Download the **Windows Installer** based on your system architecture (32-bit or 64-bit)~~

~~The LTS (Long Term Support) version is recommended for most users since it is more stable, whereas the Current version includes the latest features but may have more frequent updates.~~

Step 2: Run the Installer

- Locate the downloaded **.msi** file and double-click to run it.
- Follow the prompts in the setup wizard, accept the license agreement, and use the default settings for installation.
- Select features to install such as:
 - npm: to manage packages for Node.js applications
 - Native modules: for building native C++ modules

Step 3: Finish Setup and Install Node.js and NPM

The installer may prompt you to “install tools for native modules”. Select “Install” to complete the process.

Finish the setup

Wait for “**Finish**” to complete the setup.

Nodejs Installation

Step 4: Verify the Installation

Open **Command Prompt** or **PowerShell** > Check the installed versions by running these commands:

- Type **node -v** and press Enter to check the Node.js version.
- Type **npm -v** and press Enter to check the npm version.
- Both commands should return version numbers, confirming successful installation.

```
C:\Users\Admin> node -v
```

Note: You can run the following command, to quickly update the npm

npm install npm --global // Updates the 'CLI' client

2. Install MongoDB on Windows

For this tutorial, we'll be installing the **community edition**, which is available for free download. There's also an **enterprise edition**, but that requires a license, so we won't be dealing with it here. At the time of publication, **MongoDB 4.0.6** is the latest stable edition available for download and installation.

Download [MongoDB](#) for Windows OS release from Our Dept Lab Web Server : http://192.168.1.200/mongodb-windows-x86_64-8.0.4-signed.msi

~~Download the current stable release of **MongoDB** and install using the **Complete** setup type and all the default options.~~

Create the MongoDB data directory

Create an empty folder at C:\data\db.

MongoDB requires a directory for storing all its data, the default directory is C:\data\db, you can use a different directory if you prefer by specifying the --dbpath parameter when starting the **MongoDB** server (below).

Start MongoDB Server on Windows

Start the **MongoDB** server by running mongod.exe from the command line, mongod.exe is located in C:\Program Files\MongoDB\Server\[MONGODB VERSION]\bin. For example for **version 4.0.6** the following command will start **MongoDB**:

```
"C:\Program Files\MongoDB\Server\3.2\bin\mongod"
```

Alternatively, you can add it to your system environment PATH to call it from anywhere. The following command will append C:\Program Files\MongoDB\Server\[MONGODB VERSION]\bin\mongod to the current PATH.

```
set PATH=%PATH%;"C:\Program Files\MongoDB\Server\[MONGODB VERSION]\bin\mongod"
```

After you change the path, close and re-open the console window. You can verify it was added by running:

```
echo %PATH%
```

Once you run the mongod command, the output should look like the following if it's running:

```
2019-02-28T12:04:58.204-0500 I CONTROL [initandlisten] MongoDB starting : pid=6416 port=27017
dbpath=C:\data\db\ 64-bit host=YOUR_PC_NAME
2019-02-28T12:04:58.204-0500 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2019-02-28T12:04:58.204-0500 I CONTROL [initandlisten] db version v4.0.5
...
2019-02-28T12:04:59.070-0500 I NETWORK [initandlisten] waiting for connections on port 27017
```

3. Install Angular-CLI on Windows

Step 1: After the code editor is installed, create a new project folder. Then go to the project folder in command prompt/terminal and type below commands to create folder for frontend and backend

```
mkdir frontend
mkdir backend
```

Step 2: Navigate to the frontend folder using the command

```
cd frontend
```

Step 3: Initialize and run a Angular Project using the command

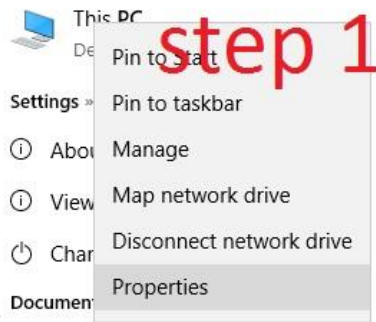
```
npm install -g @angular/cli
ng new my-angular-app
cd my-angular-app
ng serve --open
```

Step 4: Now navigate to the backend folder using the command

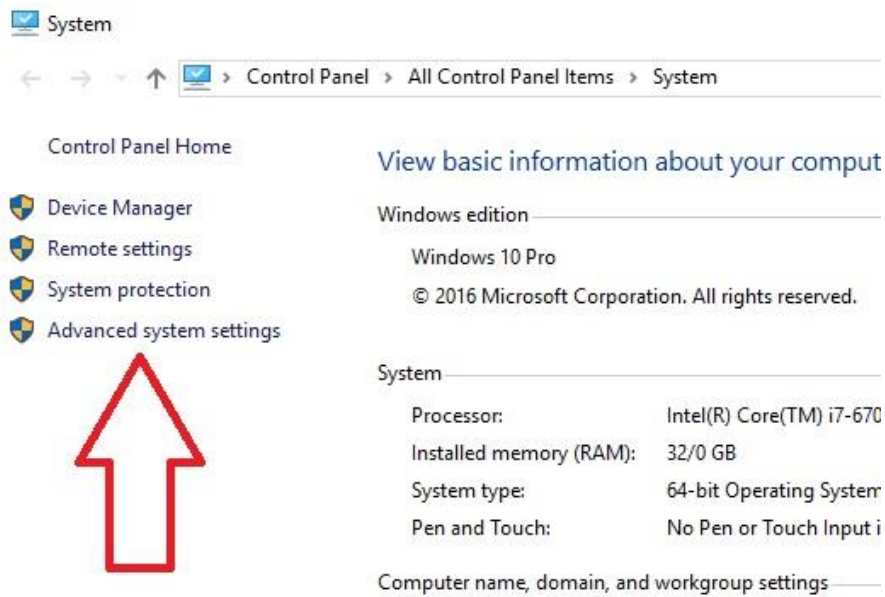
cd..

cd backend

(open in Windows 10) Control Panel\All Control Panel Items\System or accordance with the figure

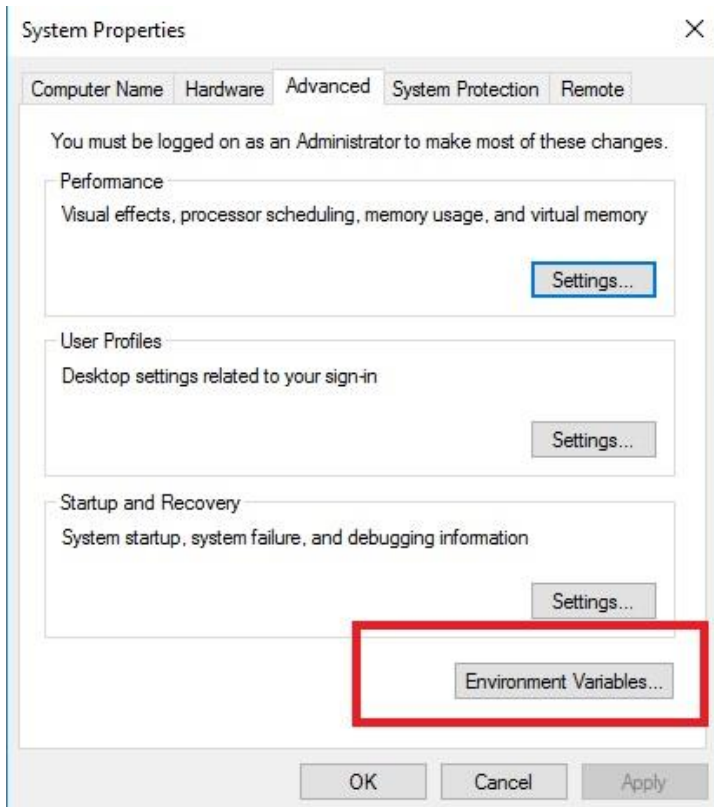


step 1:

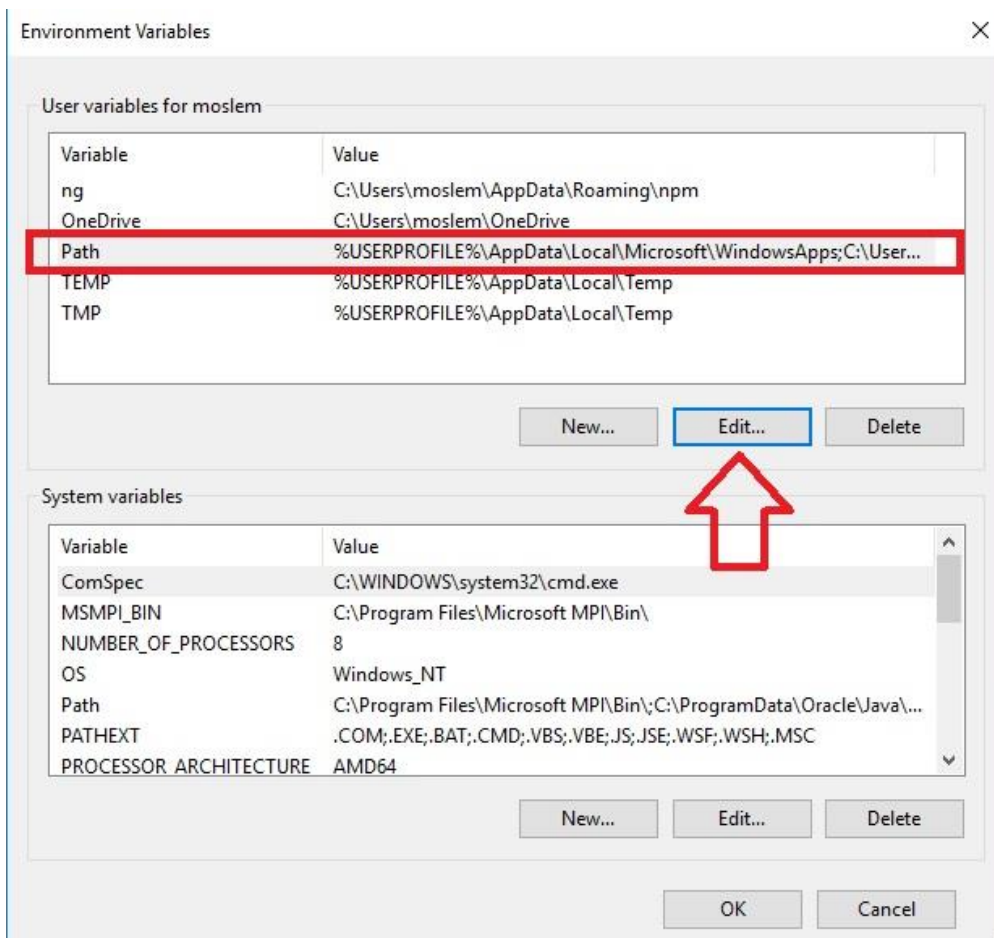


step 2 :

step3:



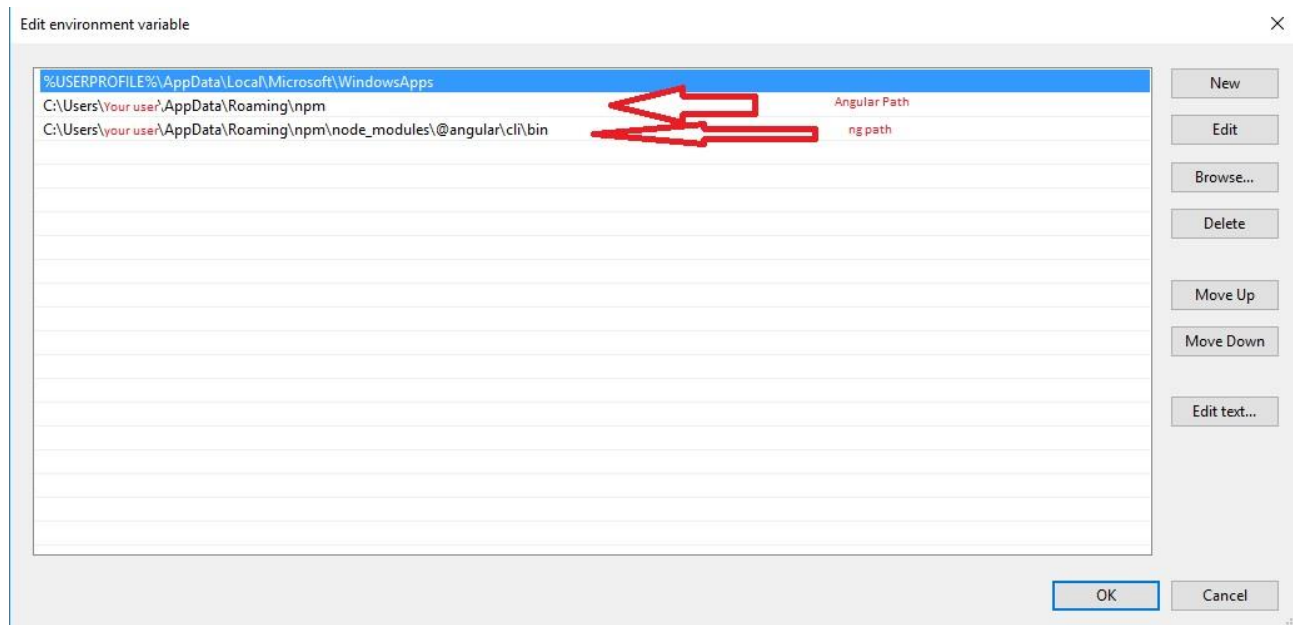
step4:



step5: add missing ng path

C:\Users\{your username}\AppData\Roaming\npm

C:\Users\{yourusername}\AppData\Roaming\npm\node_modules\@angular\cli\bin



Here is new environment variable that you need

add: C:\Users\PK\AppData\Roaming\npm\node_modules\@angular\cli\bin

Finally, restart all opened command prompts and try again.

4.Installation of Express:

Express is a lightweight web application framework for node.js used to build the back-end of web applications relatively fast and easily. Here we are going to write a simple web app that will display a message on the browser.

Setup:

- Install node: Follow the instruction given on [this](#) page if you have not installed the node.
- Check whether node and npm are installed or not by typing the following two commands in the command prompt or terminal.

```
node --version
```

```
npm --version
```

- Install Express: Make a working directory for the project .

•**STEP-1:** Creating a directory for our project and make that our working directory.

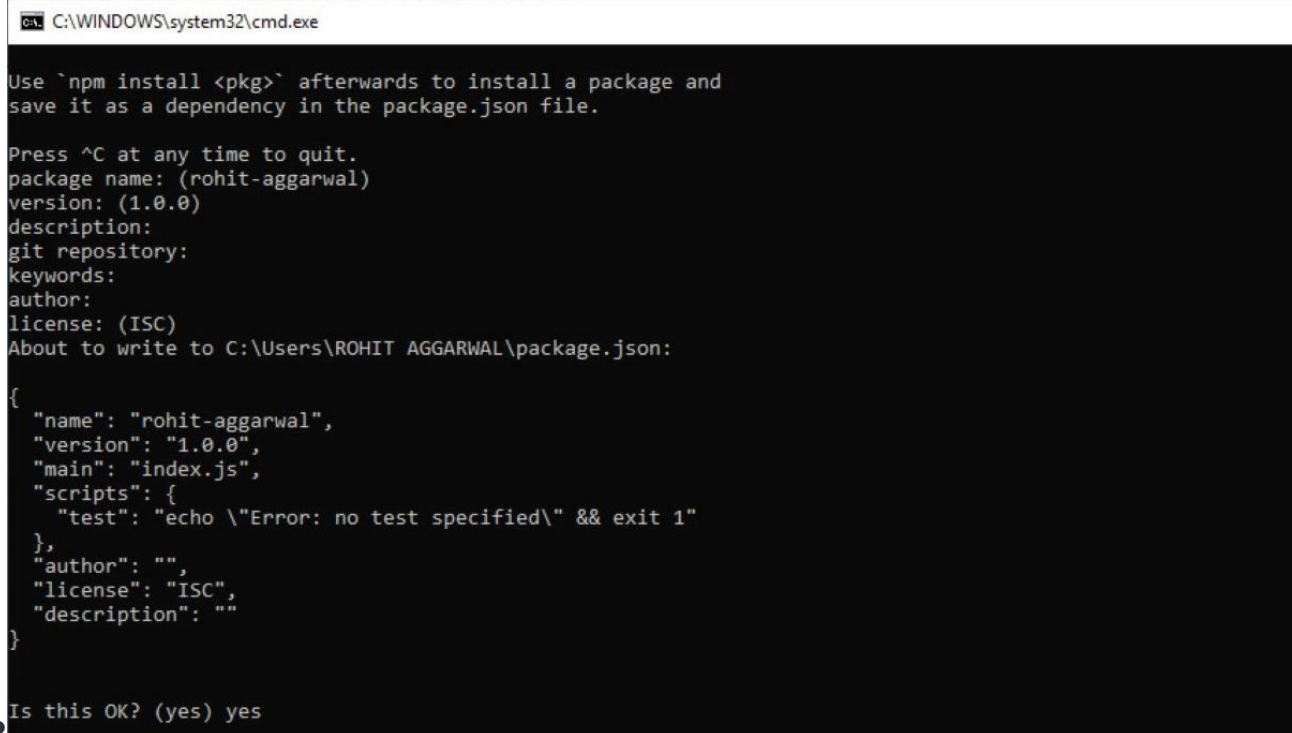
```
•$ mkdir gfg
```


- \$ cd gfg

- STEP-2:** Using npm init command to create a package.json file for our project.

- \$ npm init

- This command describes all the dependencies of our project. The file will be updated when adding further dependencies during the development process, for example when you set up your build system.



```
C:\WINDOWS\system32\cmd.exe

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (rohit-aggarwal)
version: (1.0.0)
description:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\ROHIT AGGARWAL\package.json:

{
  "name": "rohit-aggarwal",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}

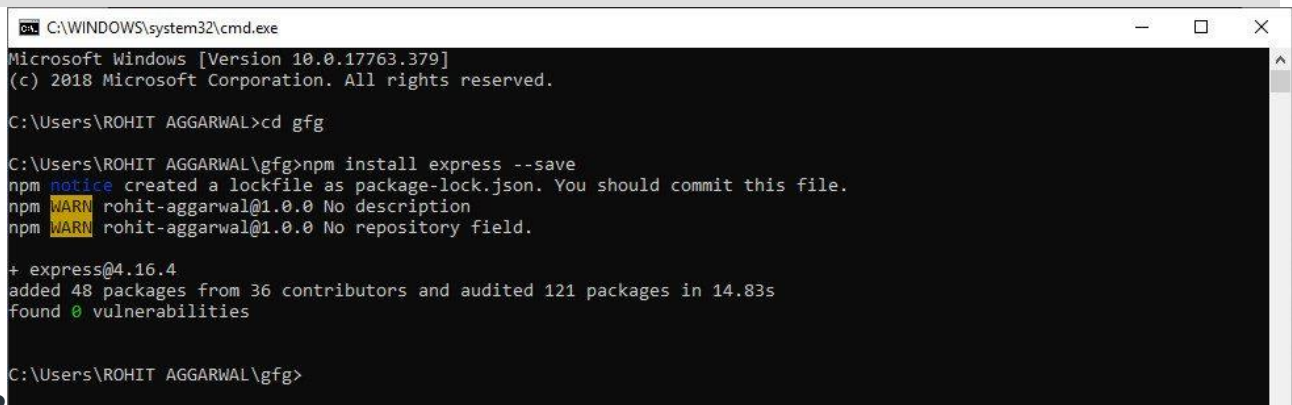
Is this OK? (yes) yes
```

- Keep pressing enter and enter “yes/no” accordingly at the terminus line.

STEP-3: Installing Express

- Now in your *gfg(name of your folder)* folder type the following command line:

- \$ npm install express --save



```
C:\WINDOWS\system32\cmd.exe

Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ROHIT AGGARWAL>cd gfg

C:\Users\ROHIT AGGARWAL\gfg>npm install express --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN rohit-aggarwal@1.0.0 No description
npm WARN rohit-aggarwal@1.0.0 No repository field.

+ express@4.16.4
added 48 packages from 36 contributors and audited 121 packages in 14.83s
found 0 vulnerabilities

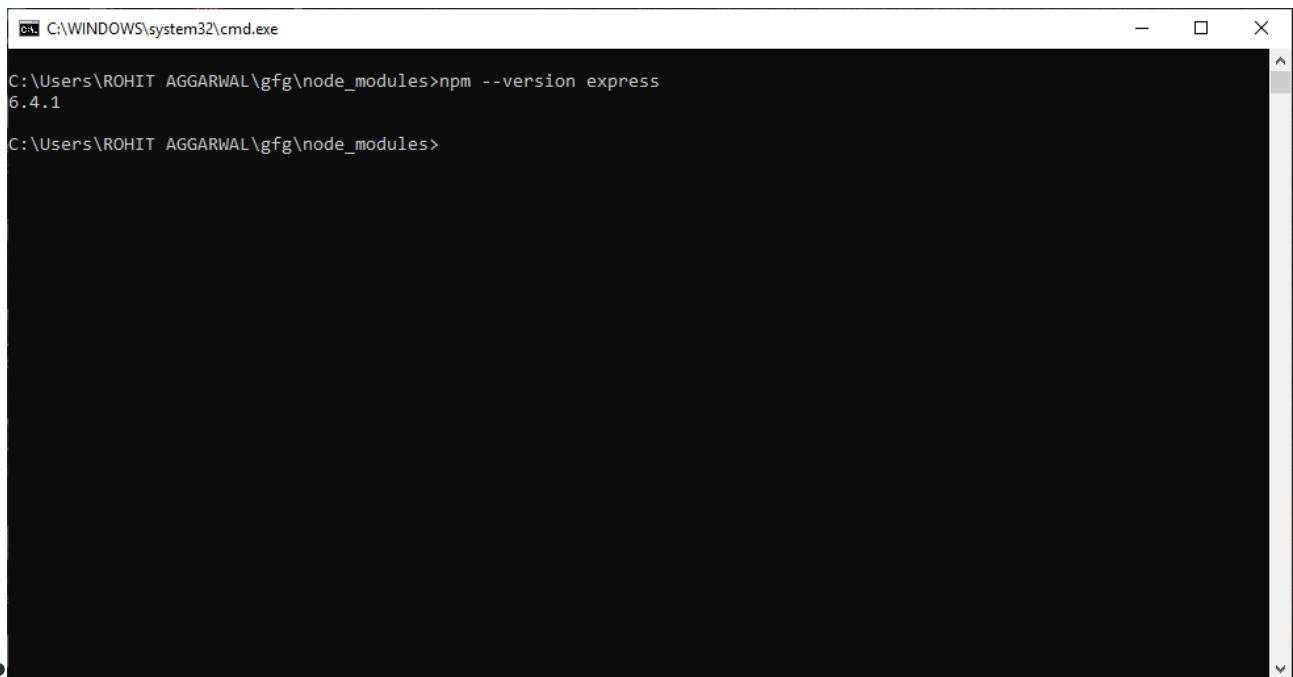
C:\Users\ROHIT AGGARWAL\gfg>
```

- NOTE-** Here “*WARN*” indicates the fields that must be entered in STEP-2.

- STEP-4:** Verify that Express.js was installed on your Windows:

- To check that express.js was installed on your system or not, you can run the following command line on cmd:

- C:\Users\Admin\gfg\node_modules>npm --version express



```
C:\WINDOWS\system32\cmd.exe
C:\Users\ROHIT AGGARWAL\gfg\node_modules>npm --version express
6.4.1
C:\Users\ROHIT AGGARWAL\gfg\node_modules>
```

The version of express.js will be displayed on successful installation.

- Create a file 'firstapp.js' and write the code as shown below.

- javascript

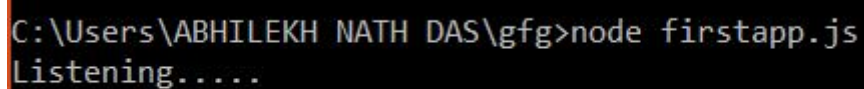
```
const express = require('express');
app = express();

app.get('/', function (req, res) {
  res.type('text/plain');
  res.status(200);
  res.send('Hi');
});

app.listen(4000, function () {
  console.log('Listening.....');
});
```

Start the app by typing the following command:

```
node filename.js
```



```
C:\Users\ABHILEKH NATH DAS\gfg>node firstapp.js
Listening.....
```

This means that the server is waiting for a request to come.

Open any browser of your choice and go to “localhost:4000/” and you will see the message “Hi”.

Explanation:

```
const express = require('express');
```

require() is a node.js function used to load the external modules. Here ‘express’ is that external module.

```
app = express();
```

Here an object of an express module is created on which different methods will be applied like get, set, post, use, etc.

```
app.get('/', function(req, res){  
  res.type('text/plain');  
  res.status(200);  
  res.send('Hi');  
});
```

get() is a function by which we add a route and a function that will get invoked when any request will come. The function which we are passing to get(), sets attributes of the response header by updating the status code as 200, mime-type as ‘text/plain’, and finally sends the message ‘GeeksforGeeks’ to the browser.

```
app.listen(4000);
```

This is used to establish a server listening at port 4000 for any request.

6. Running Your Sample MEAN Application

Let's run the sample application to make sure that the system is functioning correctly. Use npm start:dev to allow you to test your application in development mode.

```
npm start:dev
```

You may now access your **MEAN** application by visiting <http://localhost:4000> in your favorite browser.

Congrats! You've configured and run the sample application. This means you have a fully functional **MEAN stack** on your server.

MEAN STACK COMPONENTS

1. MongoDB: Cross-platform Document-Oriented Database

MongoDB is a document-oriented NoSQL database system that provides high scalability, flexibility, and performance. Unlike standard relational databases, MongoDB stores data in a JSON document structure form. This makes it easy to operate with dynamic and unstructured data and MongoDB is an open-source and cross-platform database System.

Why use MongoDB?

- Fast – Being a document-oriented database, easy to index documents. Therefore a faster response.
- Scalability – Large data can be handled by dividing it into several machines.
- Use of JavaScript – MongoDB uses JavaScript which is the biggest advantage.
- Schema Less – Any type of data in a separate document.
- Data stored in the form of JSON.
- Simple Environment Setup – Its really simple to set up MongoDB.
- Flexible Document Model – MongoDB supports document-model (tables, schemas, columns & SQL) which is faster and easier.

Creating a database: Simply done using a “use” command:

```
use database_name;
```

Creating a table: If the collection/table doesn't exist then a new collection/table will be created:

```
db.createCollection("collection_name");
```

Inserting records into the collection:

```
db.collection_name.insert  
(  
  {  
    "id" : 1,  
    "Name" : "Klaus",  
    "Department": "Technical",  
    "Organization": "HI"  
  }  
);
```

Querying a document:

```
db.collection_name.find({Name : "Klaus"}).forEach(printjson);
```

2. Express: Back-End Framework:

Express is a small framework that sits on top of Node.js's web server functionality to simplify its APIs and add helpful new features. It makes it easier to organize your application's functionality with middle ware and routing. It adds helpful utilities to Node.js's HTTP objects. It facilitates the rendering of dynamic HTTP objects.

Why use Express?

- Asynchronous and Single-threaded.
- Efficient, fast & scalable
- Has the biggest community for Node.js
- Express promotes code reusability with its built-in router.
- Robust API

Create a new folder to start your express project and type below command in the command prompt to initialize a package.json file. Accept the default settings and continue.

```
npm init
```

Then install express by typing the below command and hit enter. Now finally create a file inside the directory named app.js.

```
npm install express --save
```

Now type in the following in app.js to create a sample server.

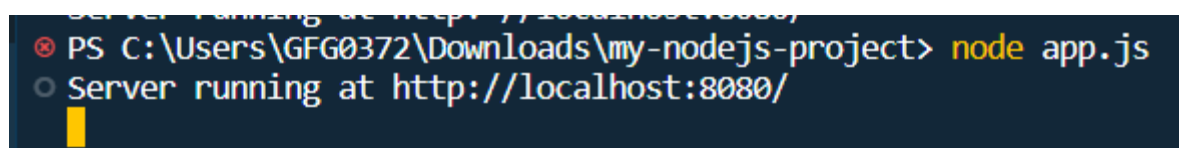
```
1
const express=require('express'),
2
http=require('http');
3
const hostname='localhost';
4
const port=8080;
5
const app=express();
6
7
app.use((req, res)=> {
8
```

```
    console.log(req.headers);
9
    res.statusCode=200;
10
    res.setHeader('Content-Type', 'text/html');
11
    res.end('<html><body><h1>This is a test server</h1></body></html>');
12
  });
13
const sample_server=http.createServer(app);
14

15
sample_server.listen(port, hostname, ()=> {
16
    console.log(`Server running at http: //${hostname}:${port}/`);});
```

Then to start the server by running the below command

node app.js

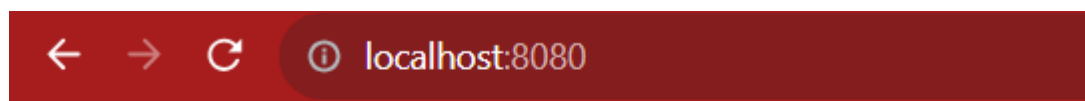


```
PS C:\Users\GFG0372\Downloads\my-nodejs-project> node app.js
Server running at http://localhost:8080/
```

Express

Terminal

Now you can open the browser and get the output of the running server.



This is a test server

Express

browser output

3. Angular JS: Open Source Frontend Framework

Angular is a Front-end Open Source Framework developed by Google Team. This framework is revised in such a way that backward compatibility is maintained (If there is any breaking change then Angular informs it very early). Angular projects are simple to create using Angular CLI (Command Line Interface) tool developed by the Angular team.

Why use Angular.JS?

- It follows a structured MVC architecture, which simplifies code organization and maintenance.
- With HTML templates and built-in directives, it simplifies complex UI tasks and data binding.
- Changes made are reflected instantly in the UI and vice versa, which reduces manual updates.
- It uses modular and reusable components, which enhances testability and maintainability.
- It utilizes TypeScript for strong typing and offers a powerful CLI for streamlined development, testing, and deployment tasks.

You can start your angular application by first installing “angular-cli” using npm or yarn.

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

After that you can create a new angular app by using.

```
ng new my-angular-app
```

Now to run the application use the following command

```
cd my-angular-app
```

```
ng serve --open
```

Terminal Output:

```
PS C:\Users\GFG0372\my-angular-app> ng serve --open
? Would you like to share pseudonymous usage data about this project with the Angular Team at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following command will disable this feature entirely:

  ng analytics disable

Global setting: enabled
Local setting: enabled
Effective status: enabled

Initial Chunk Files | Names          | Raw Size
polyfills.js        | polyfills      | 82.71 kB
main.js             | main           | 23.49 kB
styles.css          | styles         | 95 bytes
                    | Initial Total  | 106.30 kB

Application bundle generation complete. [29.355 seconds]
Watch mode enabled. Watching for file changes...
→ Local: http://localhost:4200/
```

Angular terminal output

Browser Output:



Hello, my-angular-app

Congratulations! Your app is running. 🎉

[Explore the Docs](#)

[Learn with Tutorials](#)

[CLI Docs](#)

[Angular Language Service](#)

[Angular DevTools](#)



4. Node JS: JS Runtime Environment

Node.js is used to write the Server Side Code in Javascript. One of the most important points is that it is a runtime environment which runs the JavaScript code outside the Browser. It is cross-platform and Open Source. NodeJS is not a framework and it's not a programming language. Node.js is used to build back-end services like APIs like Web App or Mobile App.

Why use Node.JS?

- Open-source JavaScript Runtime Environment
- Single threading – Follows a single-threaded model.
- Data Streaming
- Fast – Built on Google Chrome's JavaScript Engine, Node.js has a fast code execution.
- Highly Scalable

Initialize a Node.js application by typing running the below command in the command window. Accept the standard settings.

```
npm init
```

Create a file named app.js.

Example: A basic Node.js example to compute the perimeter & area of a rectangle.


```
let rectangle= {
2
  perimeter: (x, y)=> (2*(x+y)), area: (x, y)=> (x*y)
3
};
4

5
function Rectangle(l, b) {
6
  console.log("A rectangle with l = " +
7
    l + " and b = " + b);
8

9
  if (l <=0 || b <=0) {
10
    console.log("Error! Rectangle's length & "
11
      + "breadth should be greater than 0: l = "
12
        + l + ", and b = " + b);
13
  }
14

15
  else {
16
    console.log("Area of the rectangle: "
17
```

```

    + rectangle.area(l, b));
18
    console.log("Perimeter of the rectangle: "
19
    + rectangle.perimeter(l, b));
20
    }
21
}
22

23
Rectangle(1, 8);
24
Rectangle(3, 12);
25
Rectangle(-6, 3);

```

Run the node application by running the below command in the command window.

```
node app.js
```

Output:

```

PS C:\Users\GFG0372\Downloads\my-nodejs-project> node app.js
A rectangle with l = 1 and b = 8
Area of the rectangle: 8
Perimeter of the rectangle: 18
A rectangle with l = 3 and b = 12
Area of the rectangle: 36
Perimeter of the rectangle: 30
A rectangle with l = -6 and b = 3
Error! Rectangle's length & breadth should be greater than 0: l = -6, and b = 3
PS C:\Users\GFG0372\Downloads\my-nodejs-project>

```

Node terminal Output

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

After that you can create a new angular app by using.

```
ng new my-angular-app
```

Now to run the application use the following command

```
cd my-angular-app
ng serve --open
```

Terminal Output:

```
PS C:\Users\GFG0372\my-angular-app> ng serve --open
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

  ng analytics disable

Global setting: enabled
Local setting: enabled
Effective status: enabled

Initial Chunk Files | Names          | Raw Size
polyfills.js        | polyfills     | 82.71 kB |
main.js             | main          | 23.49 kB |
styles.css          | styles        | 95 bytes |

| Initial Total | 106.30 kB

Application bundle generation complete. [29.355 seconds]
Watch mode enabled. Watching for file changes...
→ Local: http://localhost:4200/
```