# **Responsive Dynamics Website using HTML, B5, NodeJS, ExpressJS, MongoDB & Mongoose**

Mainly we require to software

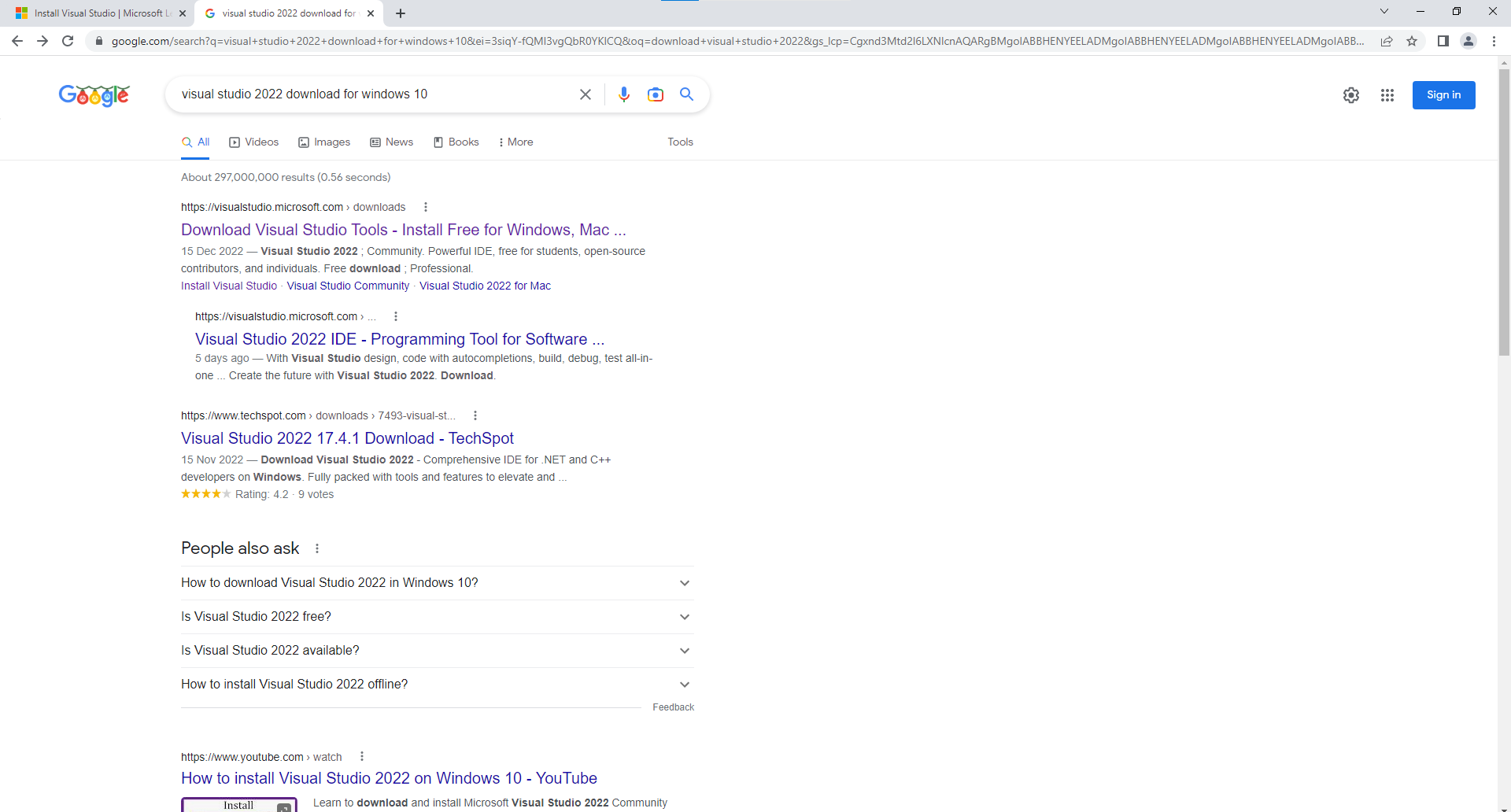
* Visual studio
* Mongodb

First we should install visual Studio:

**How to install Visual Studio**

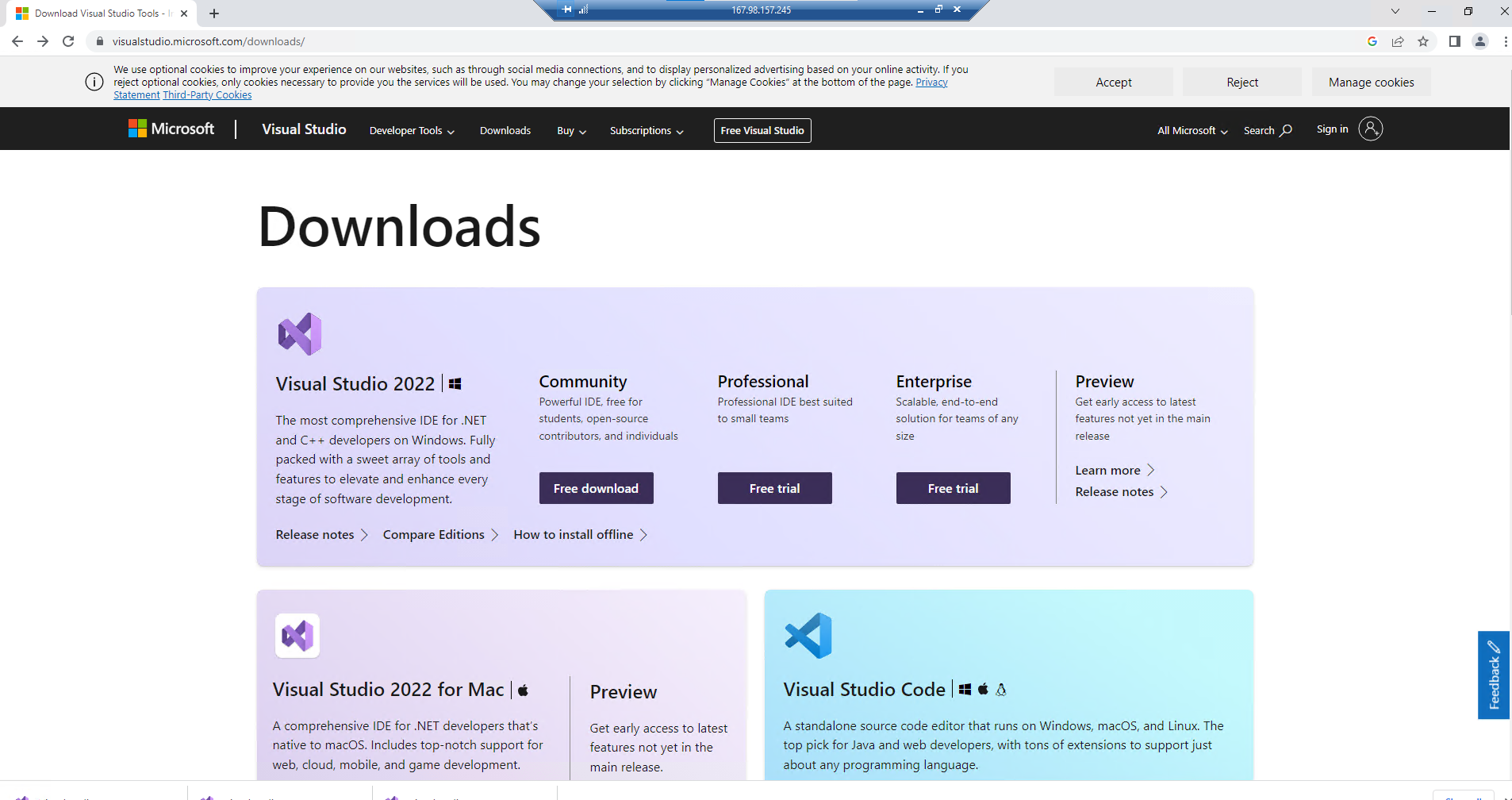
Step1: Download Visual Studio

Step2: Double click the downloaded file

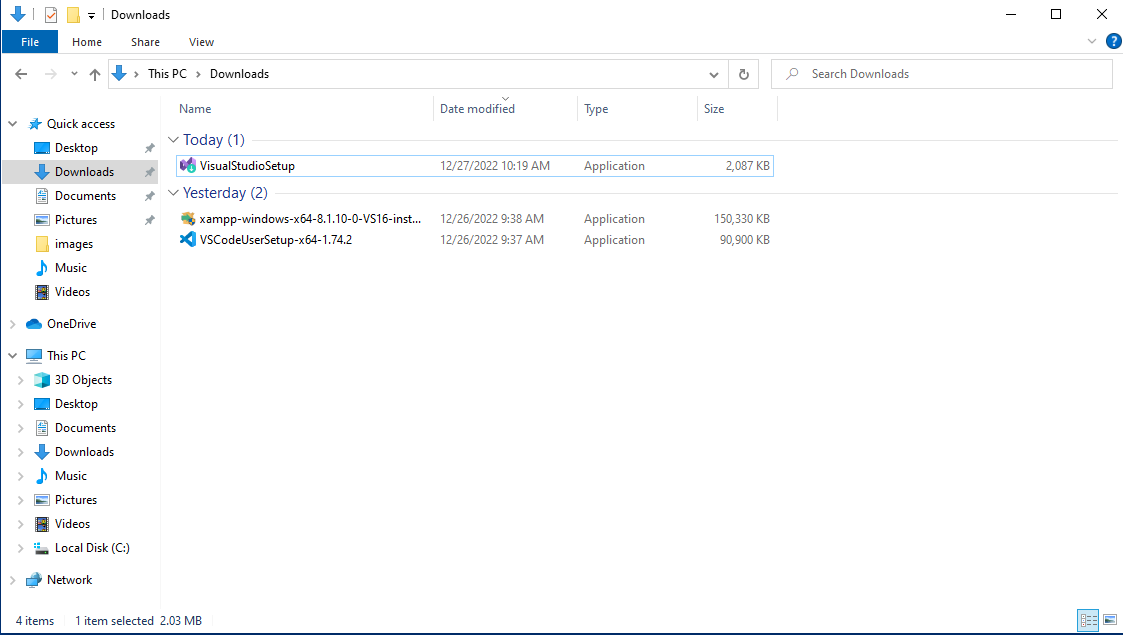


Click the first link

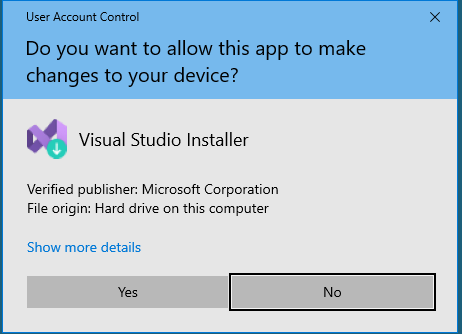
Step3: Click **free download** from **Community**.



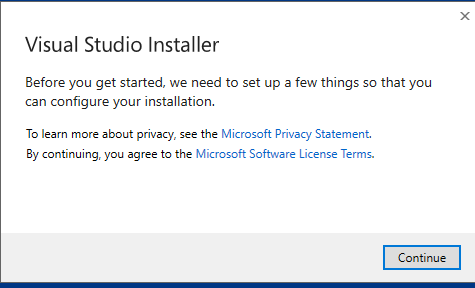
Step4: Double click the downloaded file from downloads.



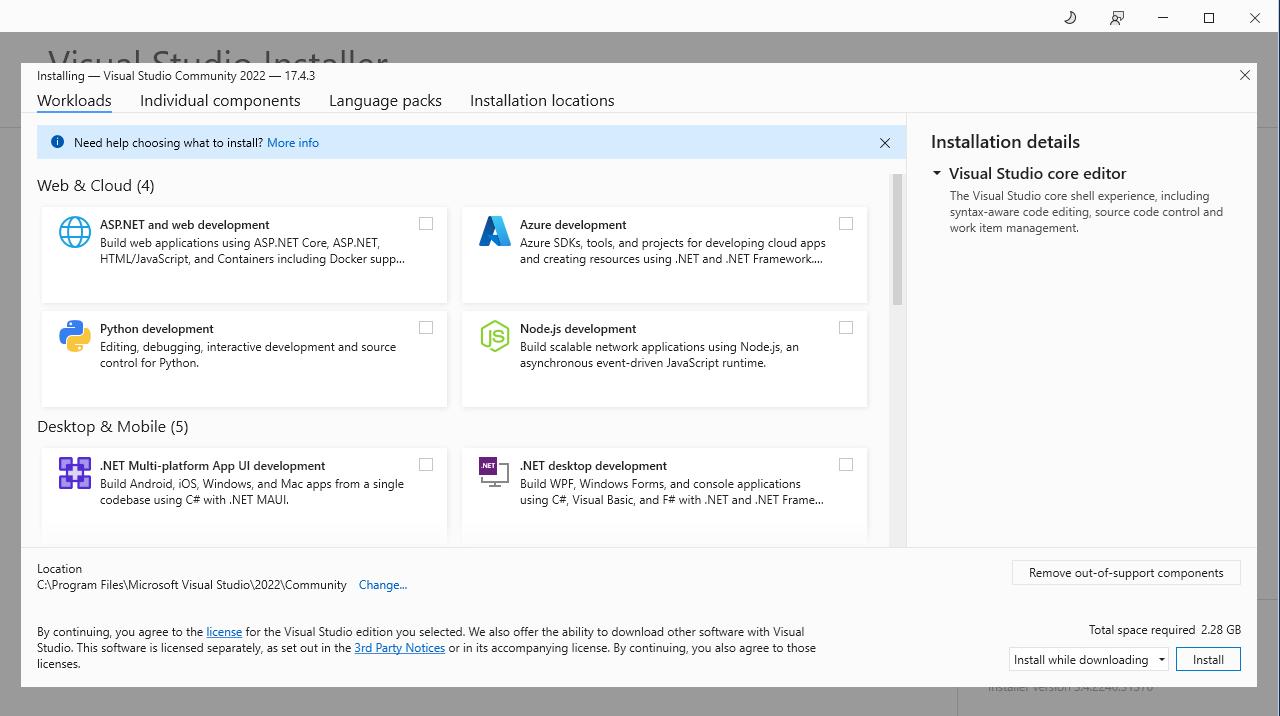
Step5: Then click ’yes’



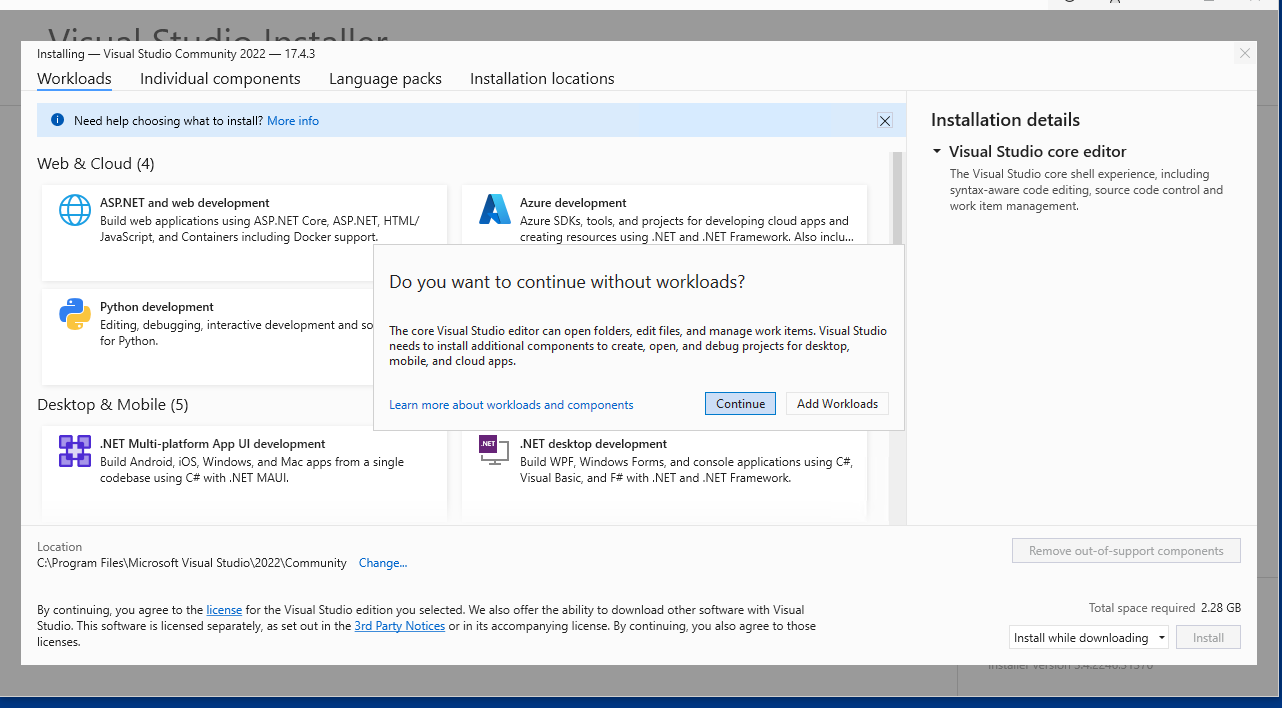
Step6: Click continue.



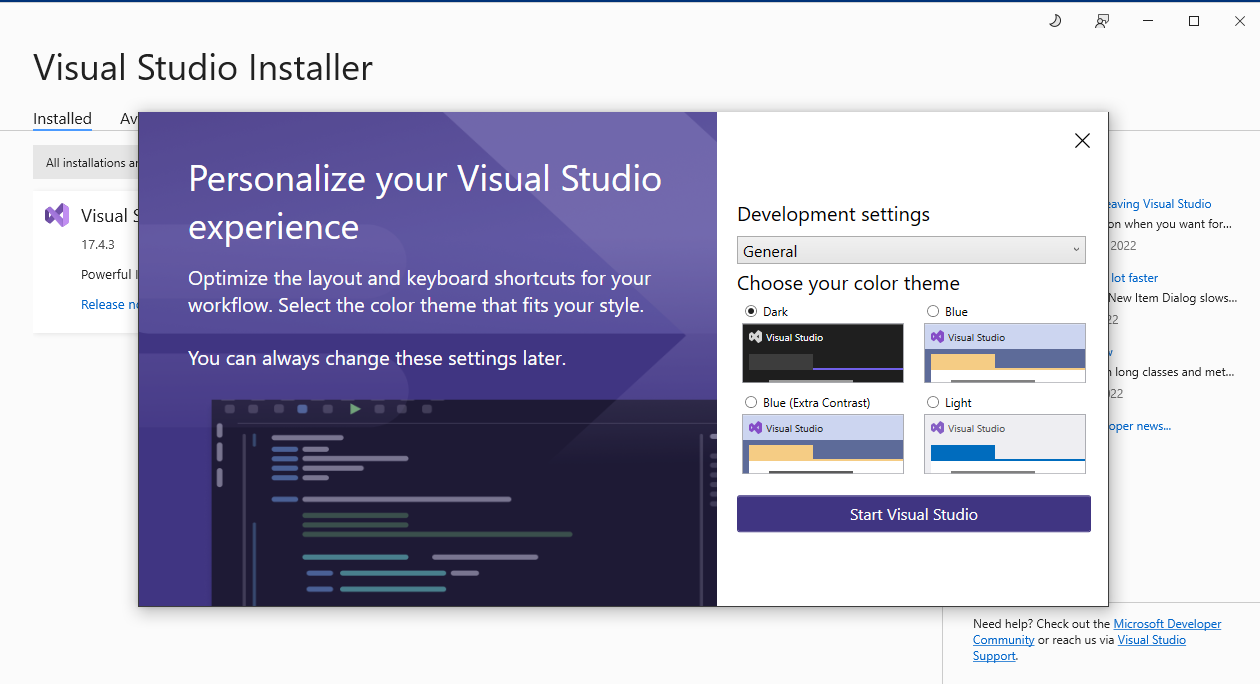
Step7: Click the install button



Step8: Click the continue button



Step9: Then click **start visual studio**



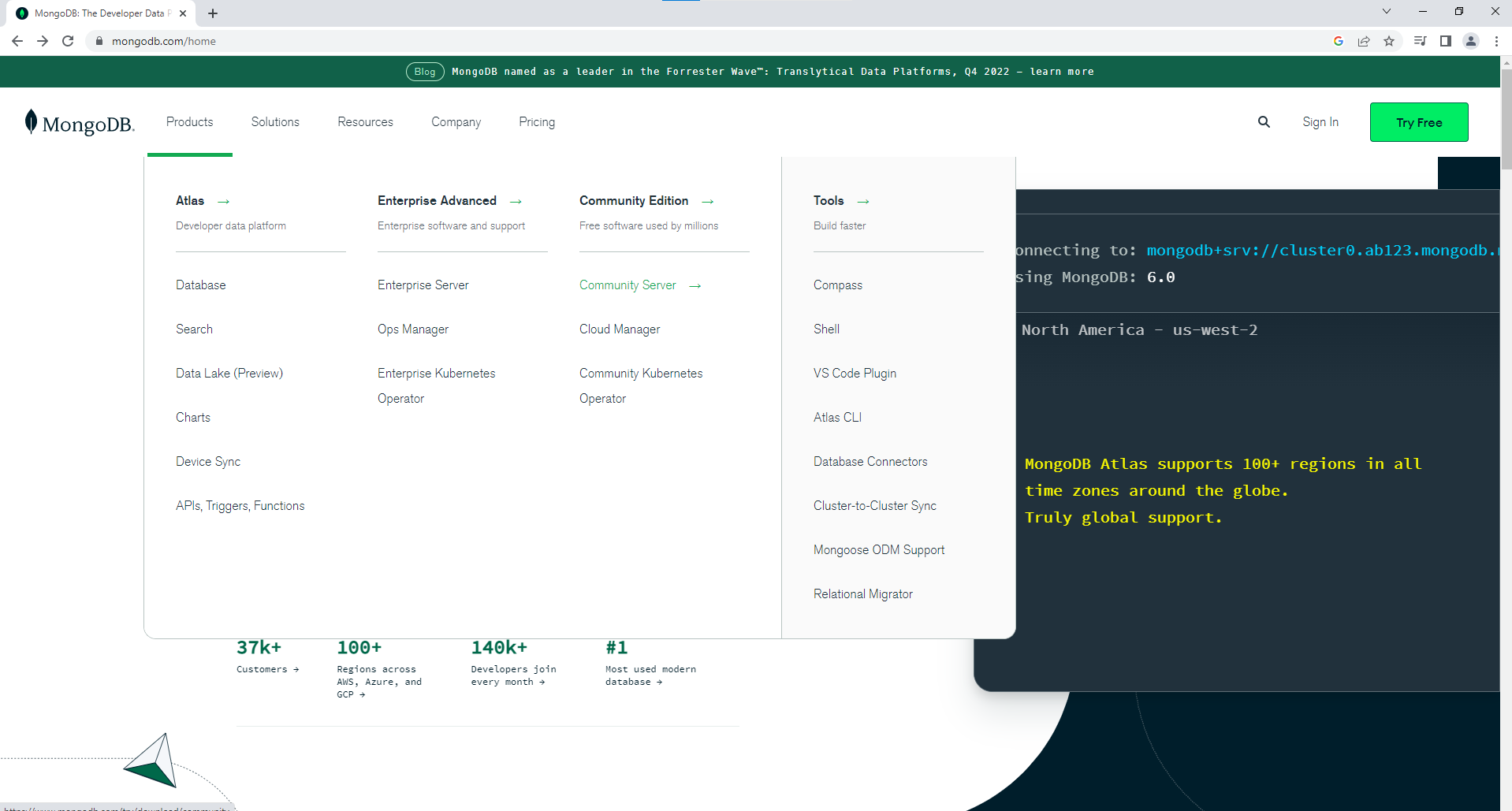
Now visual studio will appear in your screen.

**Steps for installing MongoDB in Window10**

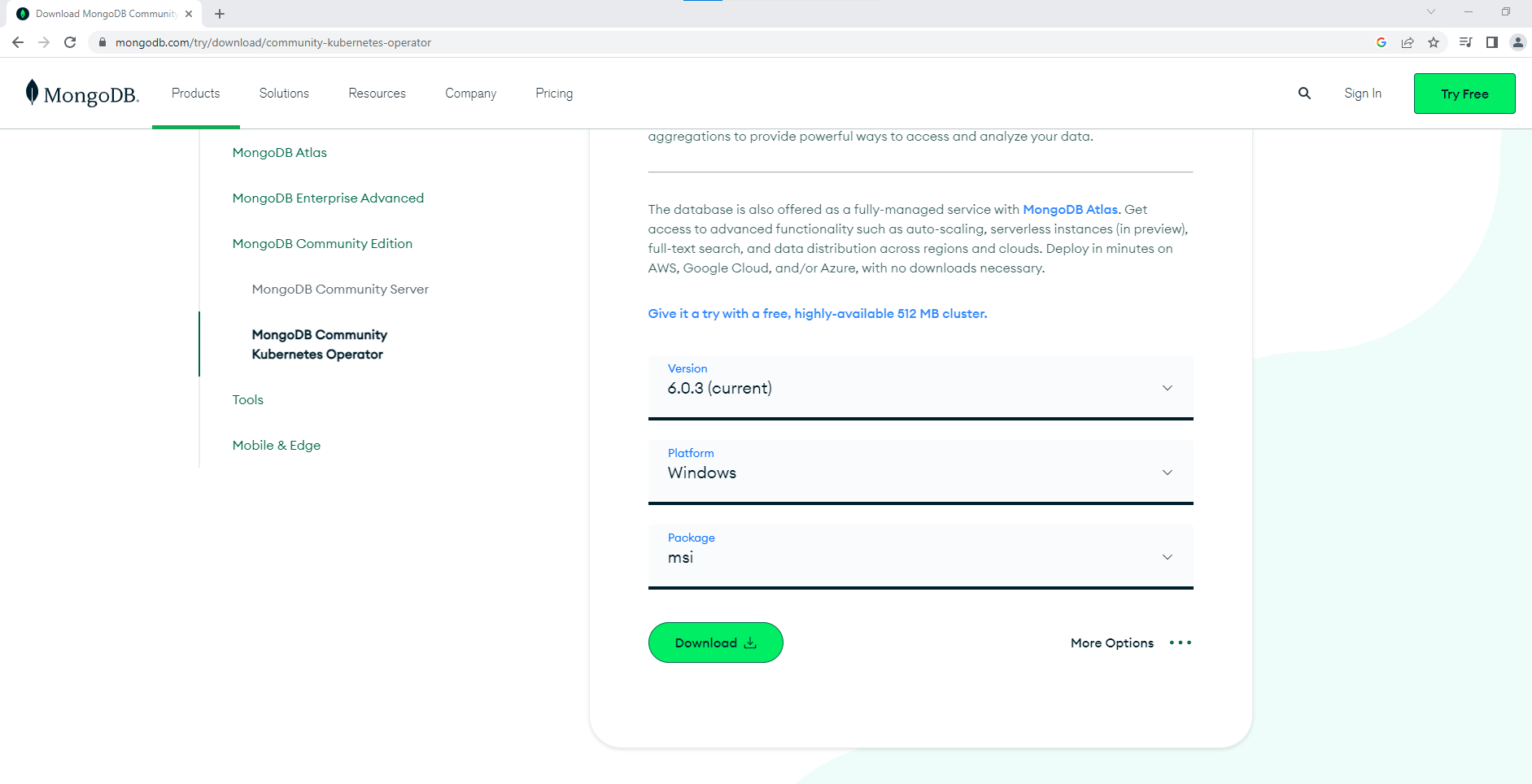
1. Search Mongodb in google and click the following link

<https://www.mongodb.com/>

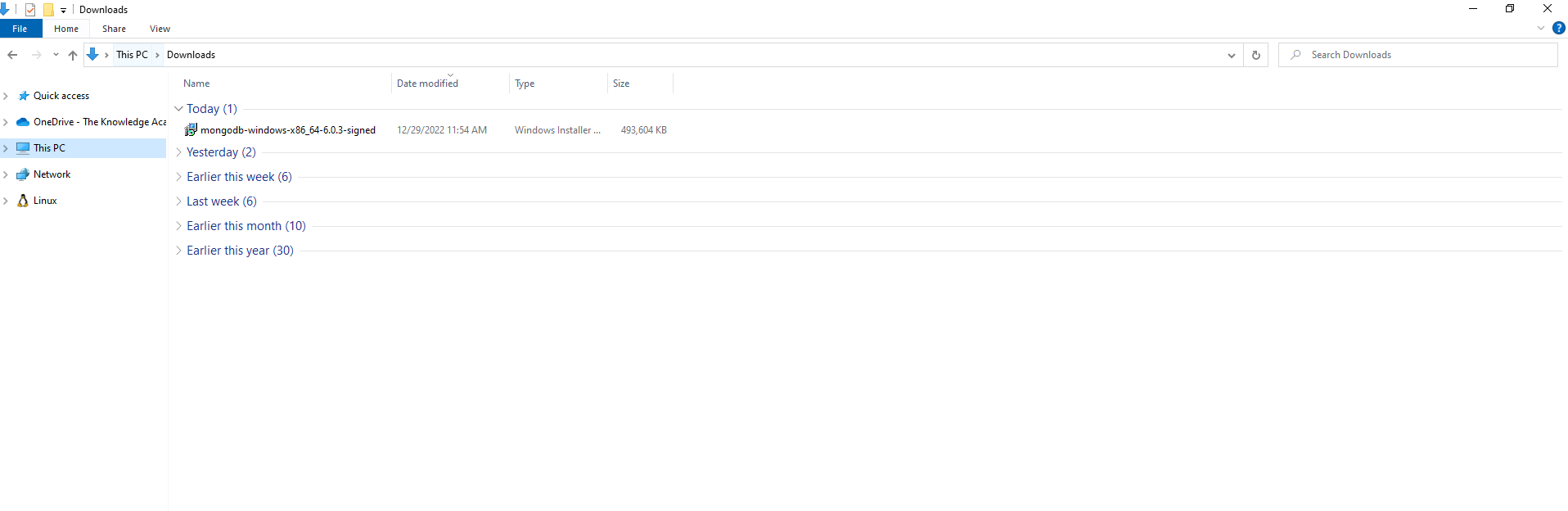
1. Click Products->Community Edition->Community Server



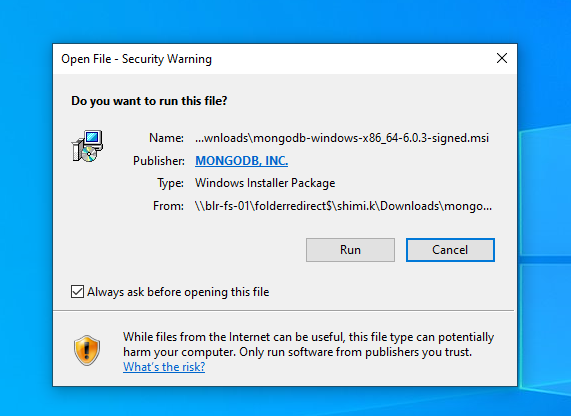
1. Click the **Download** button



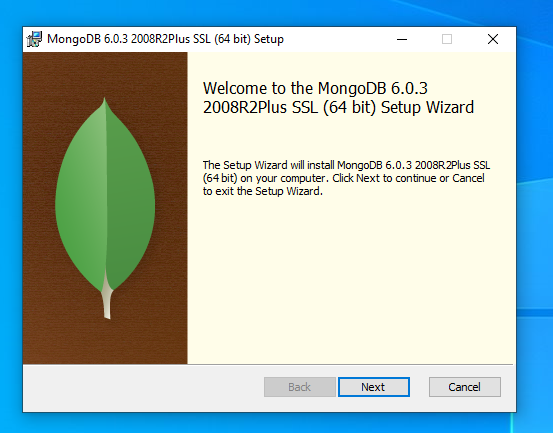
In downloads we can see the windows installer package



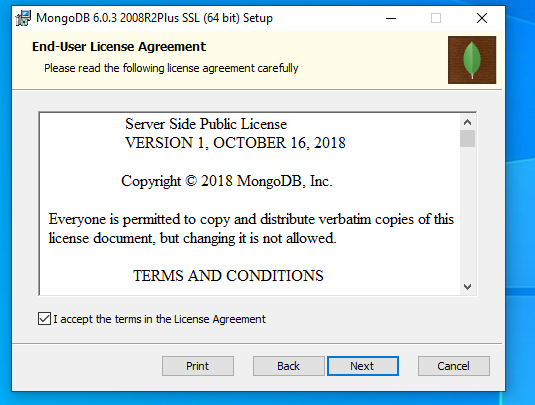
1. Double click the installer package. Then click the run button



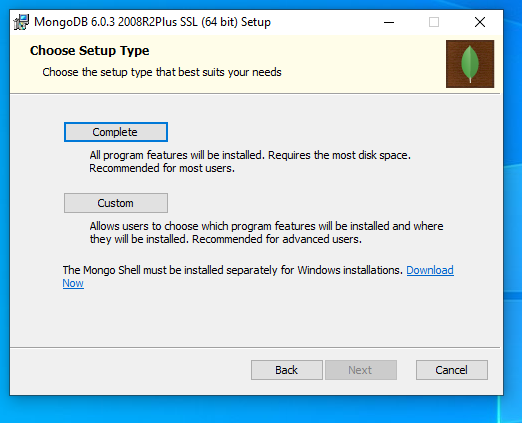
1. Click the next button



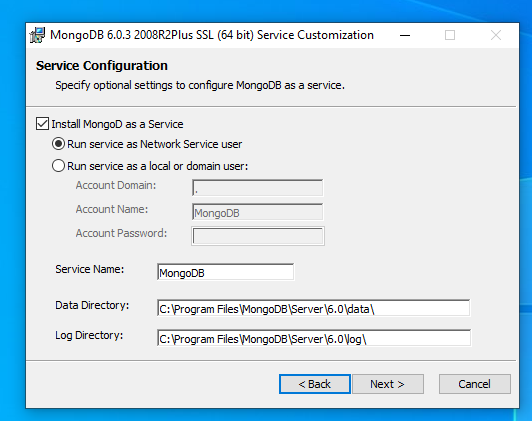
1. Tick the checkbox and click next button



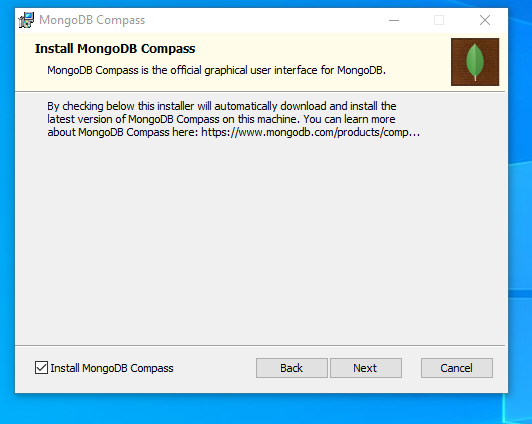
1. Click the complete option button



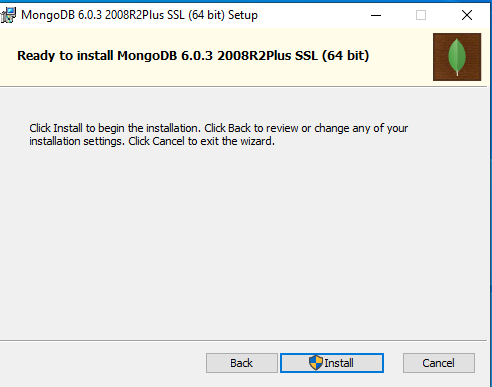
1. Select the default options and click the next button



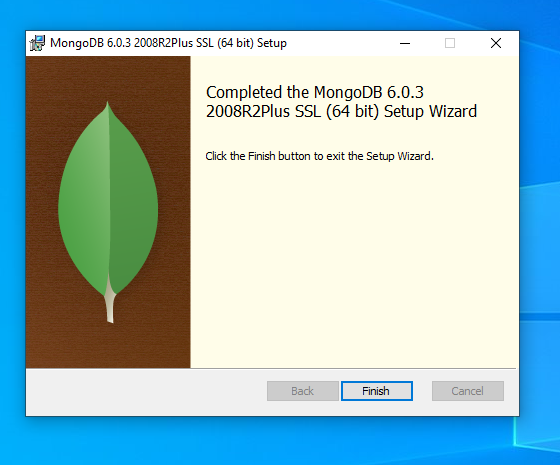
1. Click the next button



1. Click the install button



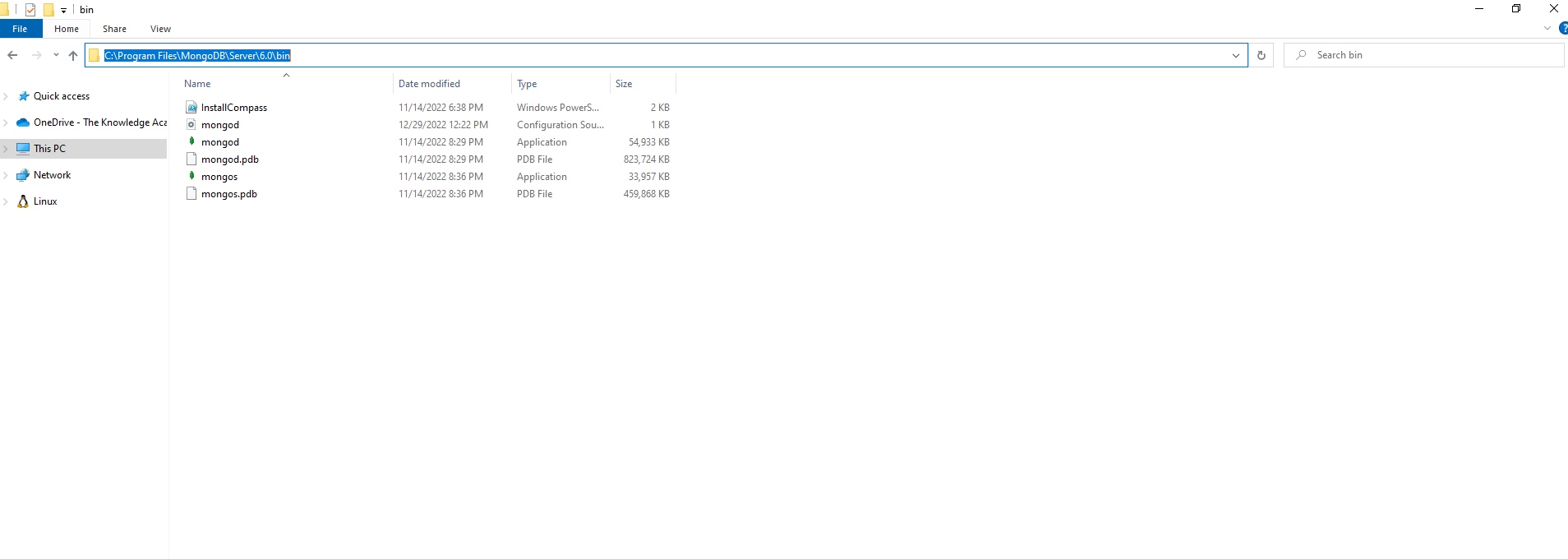
1. Click the finish button.



Finally, our mongodb is installed in our system.

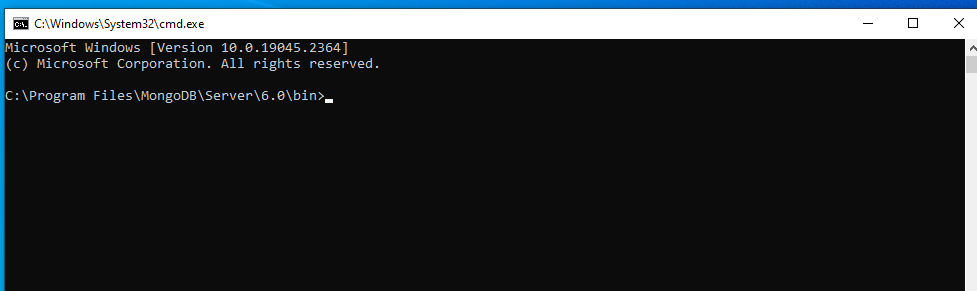
After installation go to the mongodb installed directory. Go to the following directory

C:\Program Files\MongoDB\Server\6.0\bin.

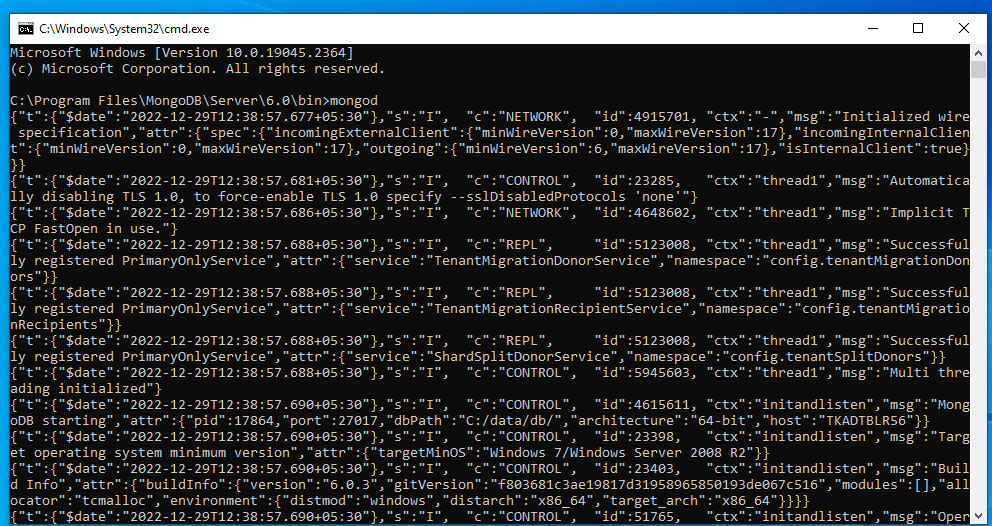


From there open cmd. Then write the following in cmd.

**cd C:\Program Files\MongoDB\Server\6.0\bin**

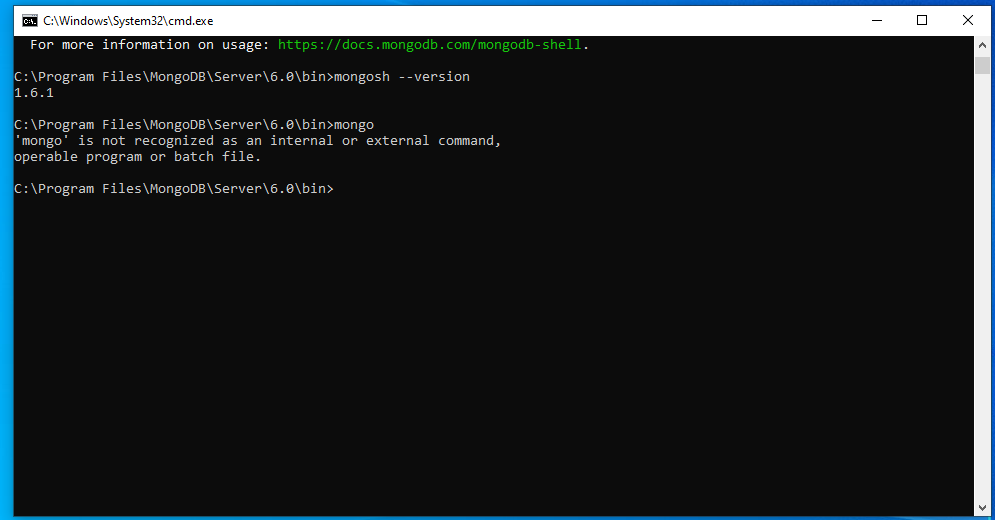


1. Write **mongod** command in cmd.



Now mongo daemon has been started.

1. Open one more cmd in same path and type **mongo.** It shows the following

****

It shows ‘'mongo' is not recognized as an internal or external command,

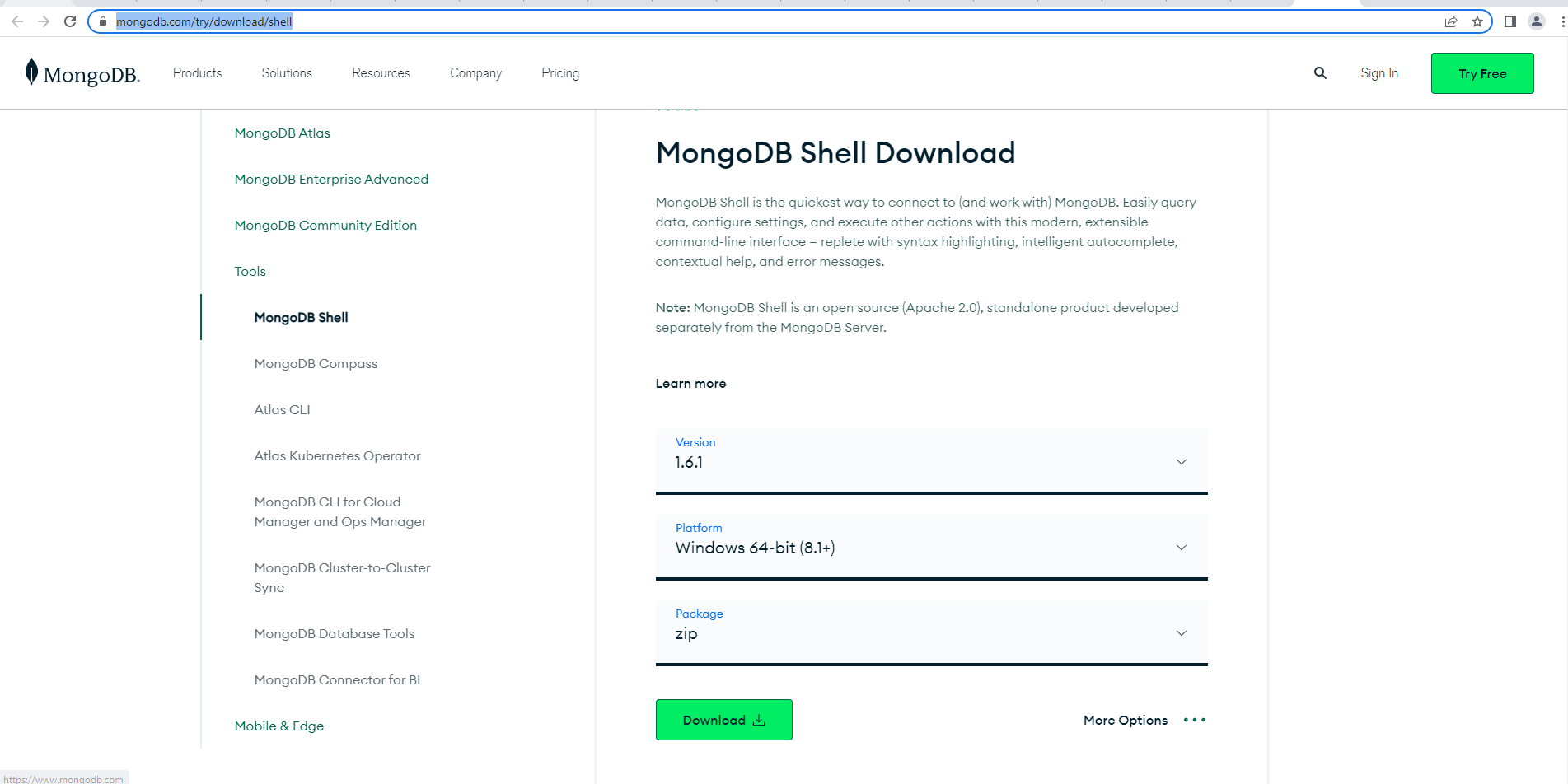
operable program or batch file.’ Because mongo will not work in maongodb version >= 6.0.

Instead of **mongo** we can use **mongosh**.

So we should install **mongosh** in our system.

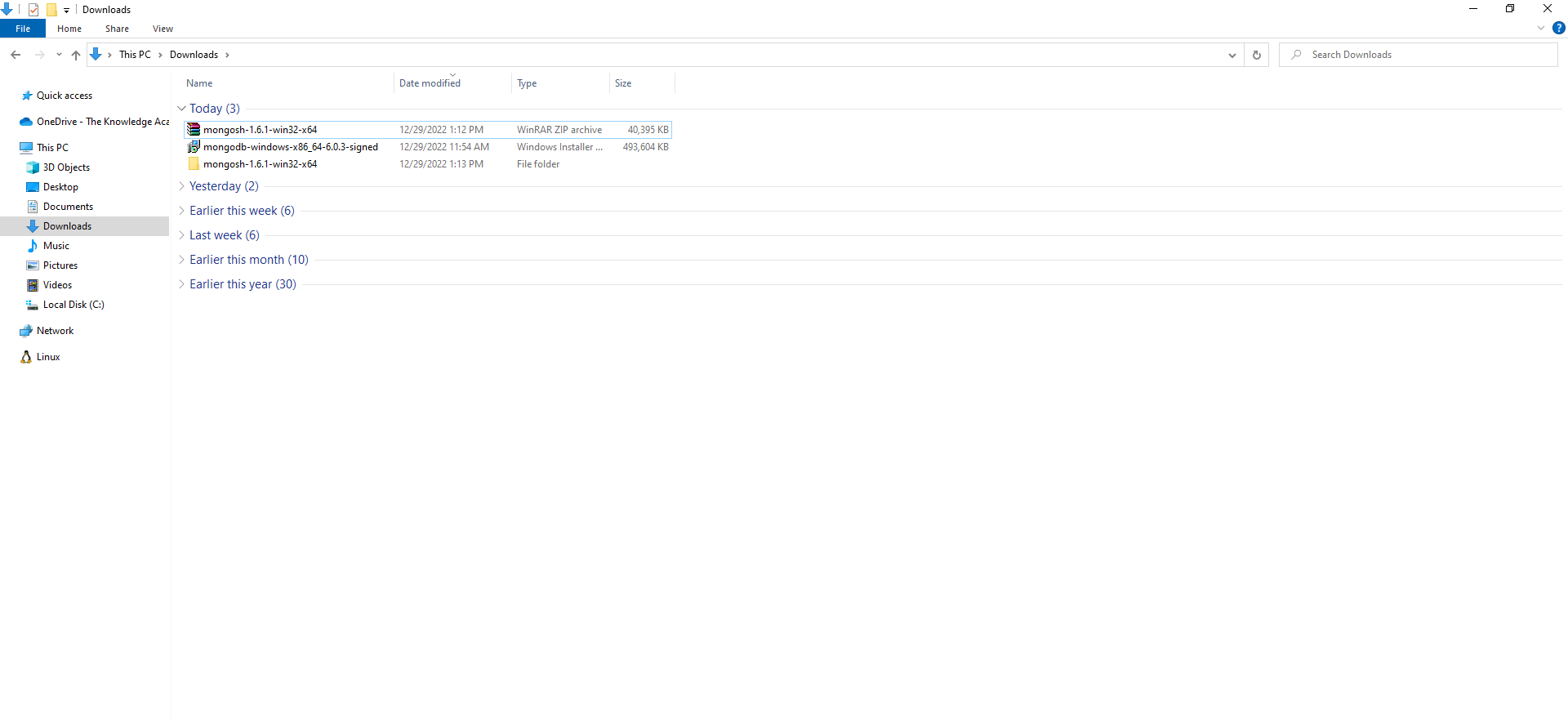
#### First Open the MongoDB Shell download page.

Open the <https://www.mongodb.com/try/download/shell>



Click the download button

Then in downloads we can see as following



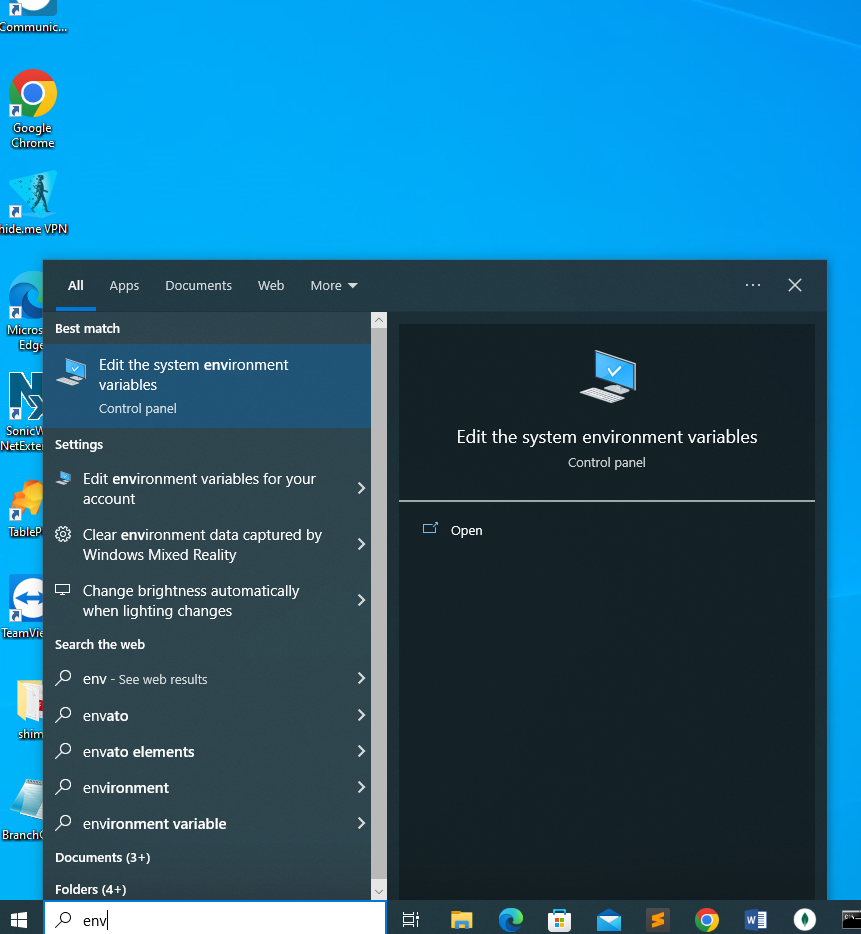
Extract the zip file.

Then open this folder **Downloads\mongosh-1.6.1-win32-x64\mongosh-1.6.1-win32-x64\bin** and copy the location.

Then set environment variable for mongosh

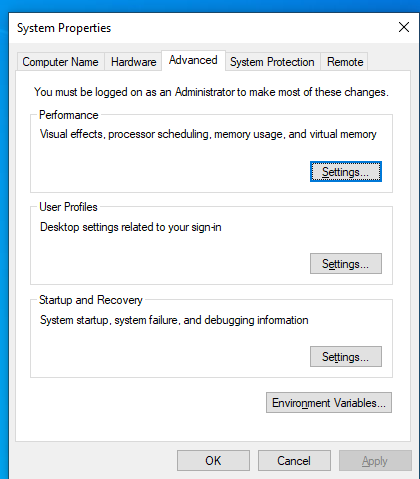
**set environment variable for mongosh**

1. search **env** in start menu.

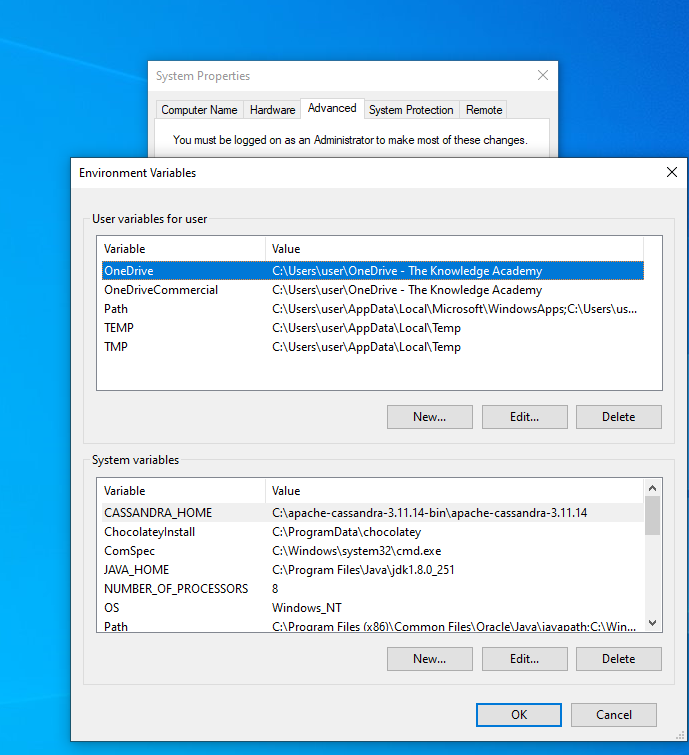


1. Click the **edit the system environment variables.** Then we can see the following window. Then

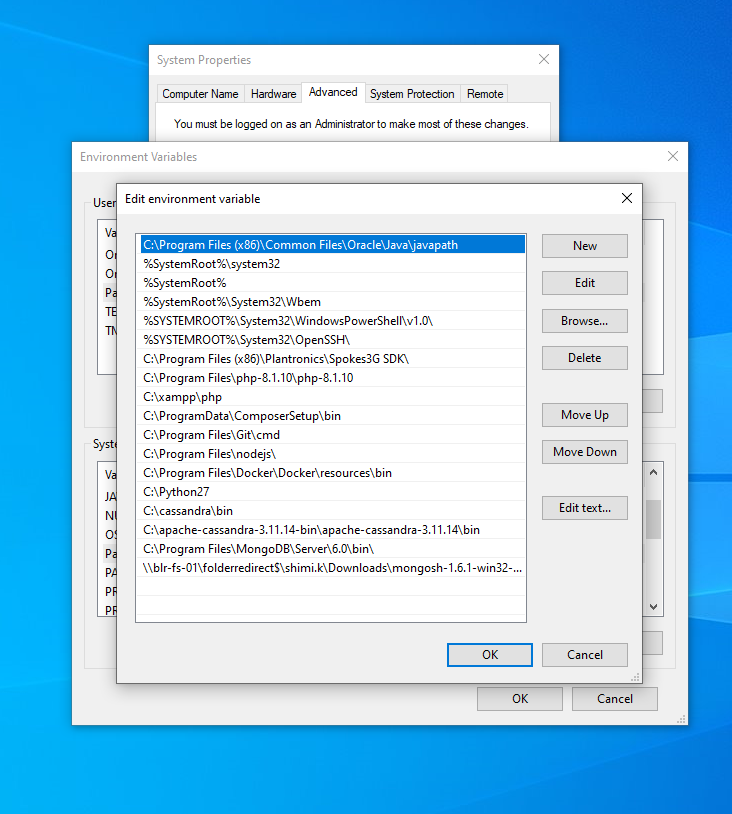
click the environment variable button.

****

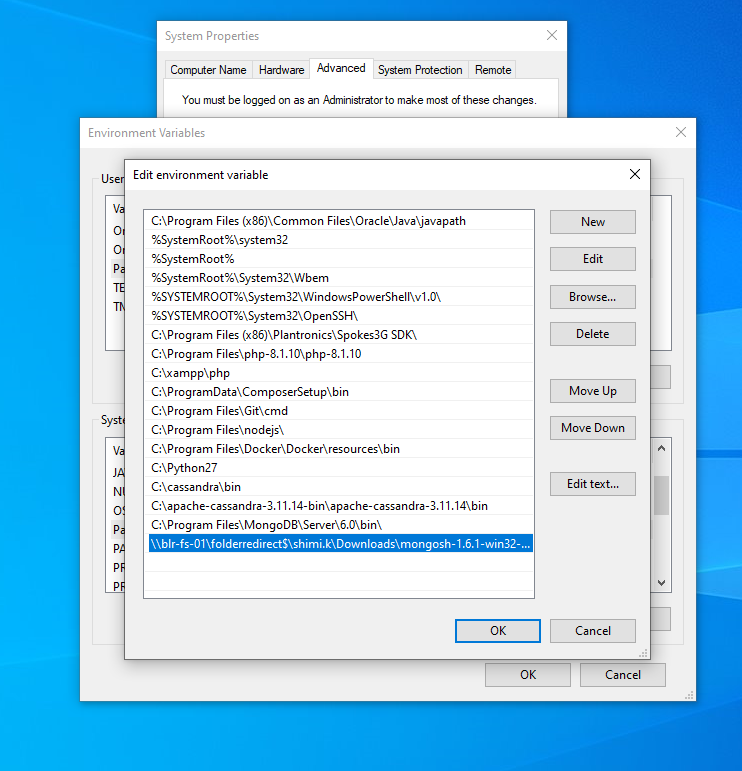
1. Click **edit** button from system variable.



After we can see the following window.



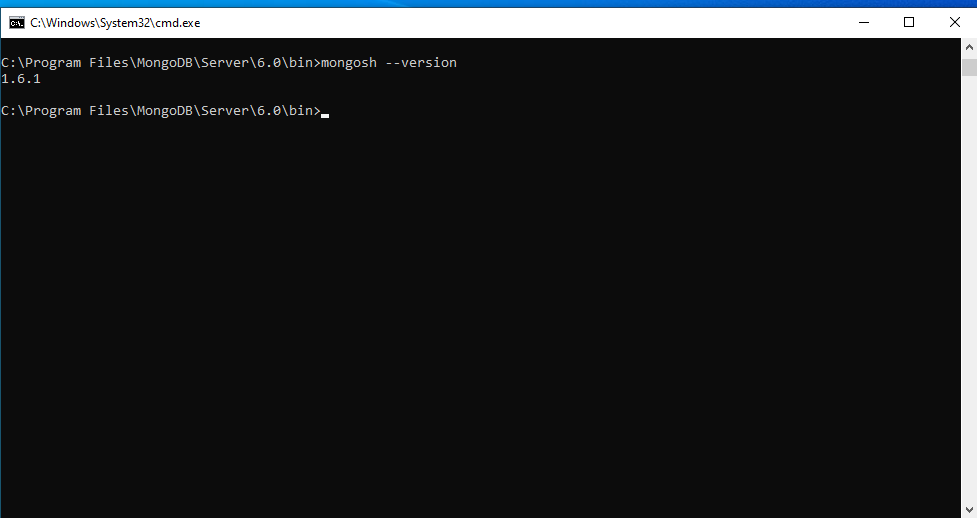
From the above window click the new button and paste the location of bin in mongosh



Then click **ok** ins each window.

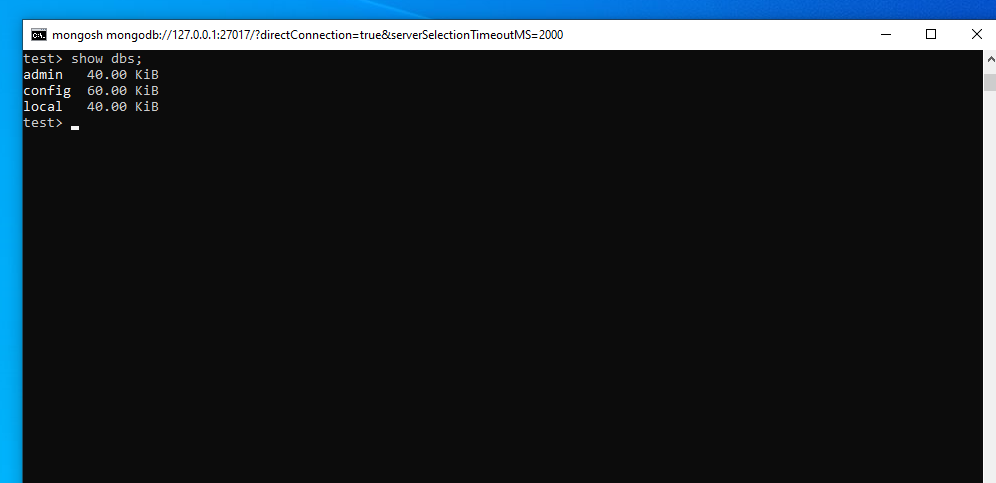
Finally, environment variable setup is ready.

We can check this by opening **cmd** and check **version of mongosh.**

****

The following command is used to check the default databases in mongodb.

**show dbs;**



Now the mongodb installation is completed.

**Database Creation:**

**Two ways** we can create database.

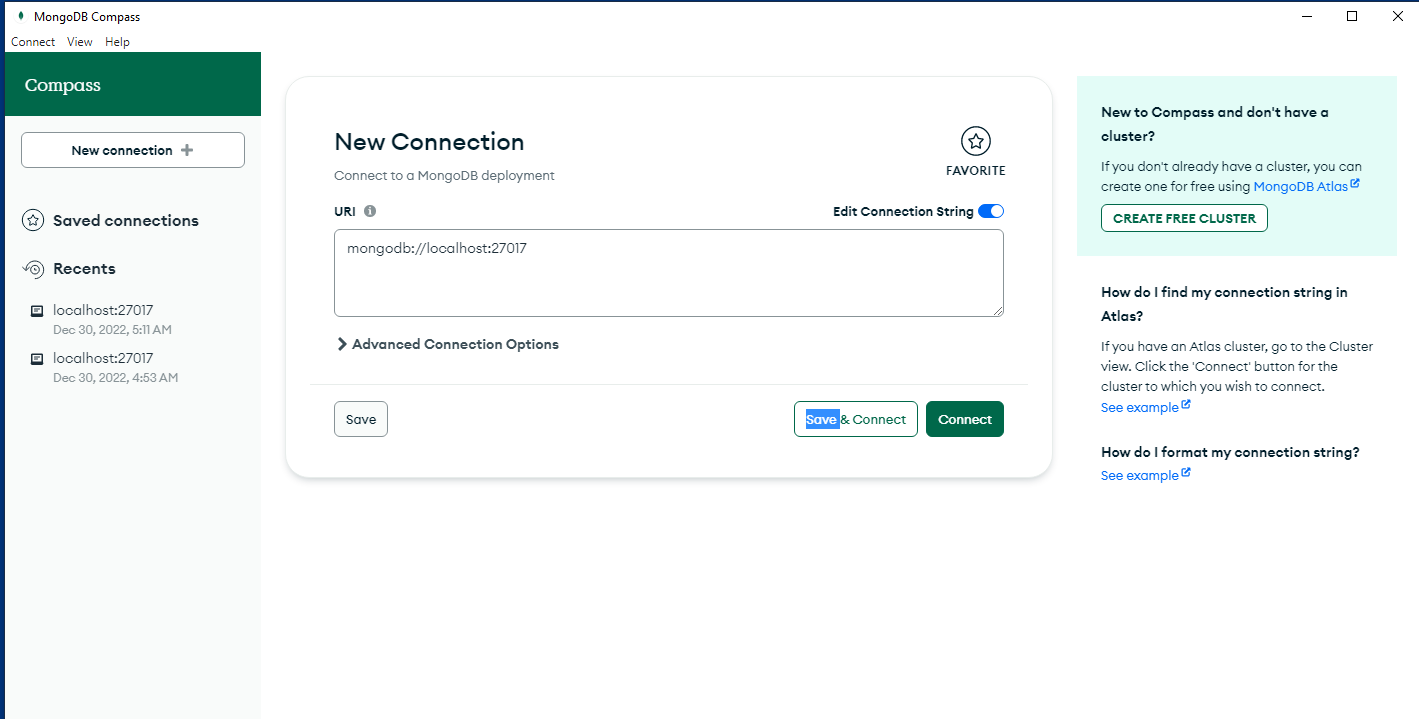
* Using command prompt
* Using GUI for mongo DB

I used the second option. It is user friendly.

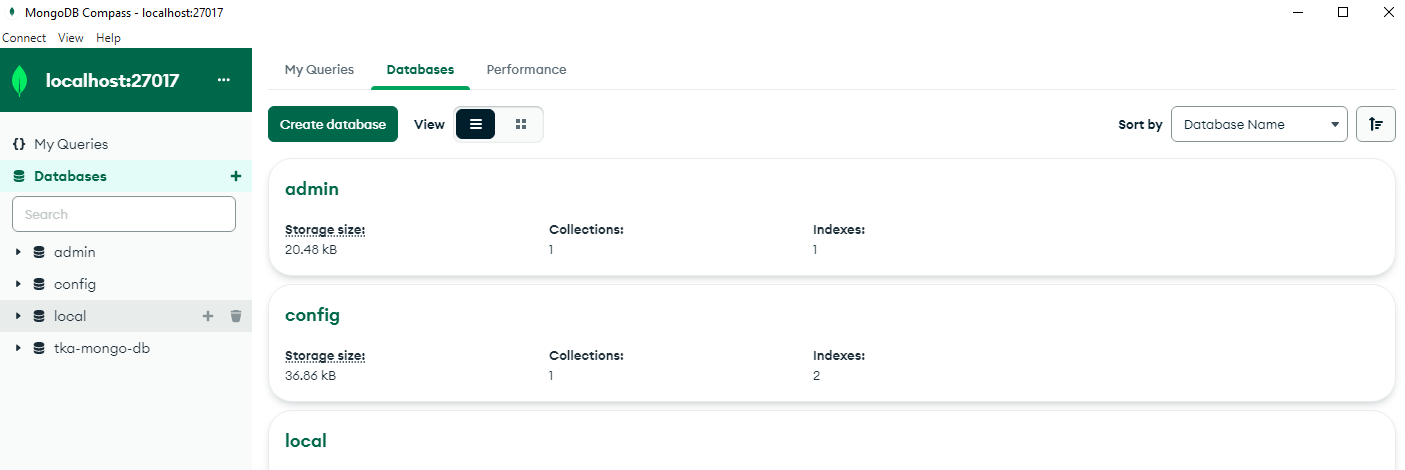
**Steps for creating database in mongoDB:**

**MongoDB Compass GUI:**

1. In our desktop we can see the **mongo DB Compass** software. Double click that one. It will open the following window. On that window click the connect button.



1. Then we can see the following window.

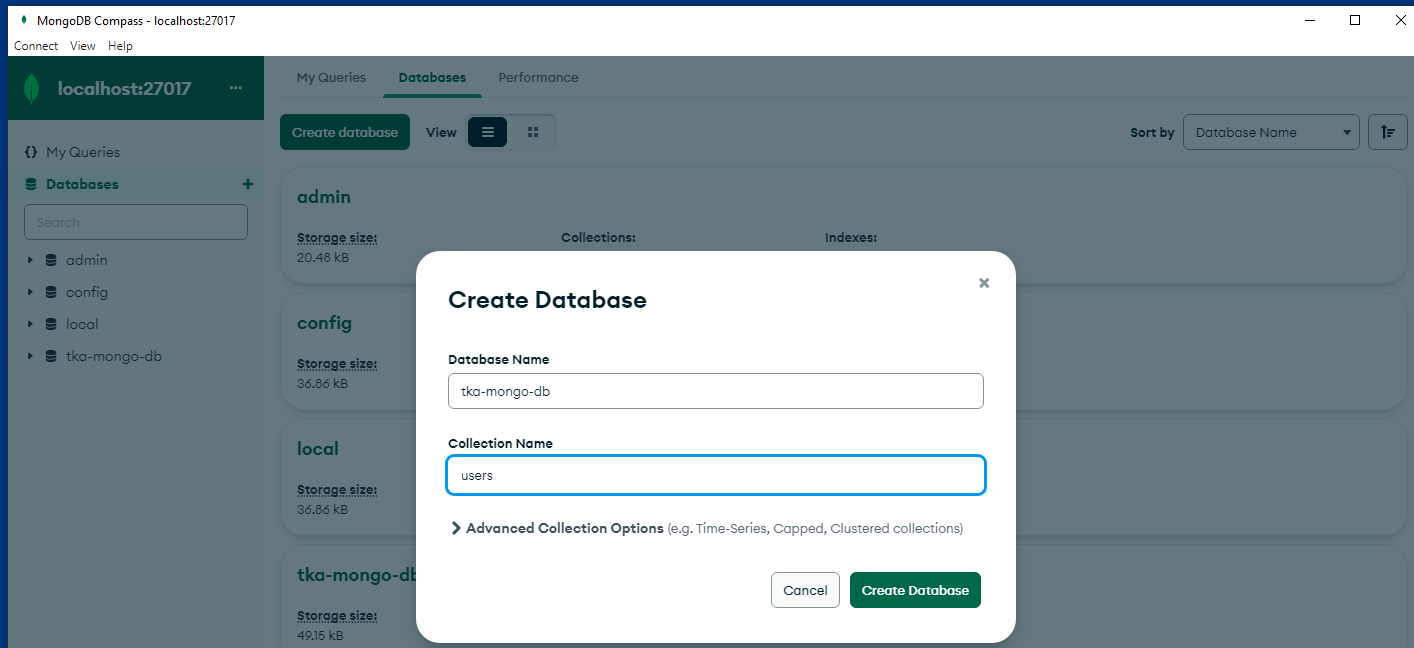


In mongoDBwe can see the 3 default databases. They are admin, config,

and local. For creating new database Click the **Create database** button.

1. On that window we can add the database name and collection name. Our database name is **tka-mongo-db** and collection name is **users**.

Collection name is same as table name in sql.

****

It will create database in mongodb.

**Website Creation:**

1. Create a folder **tka-mongo-html-website** anywhere in your system.
2. Open the folder from visual studio.
3. Next Step is to type the command

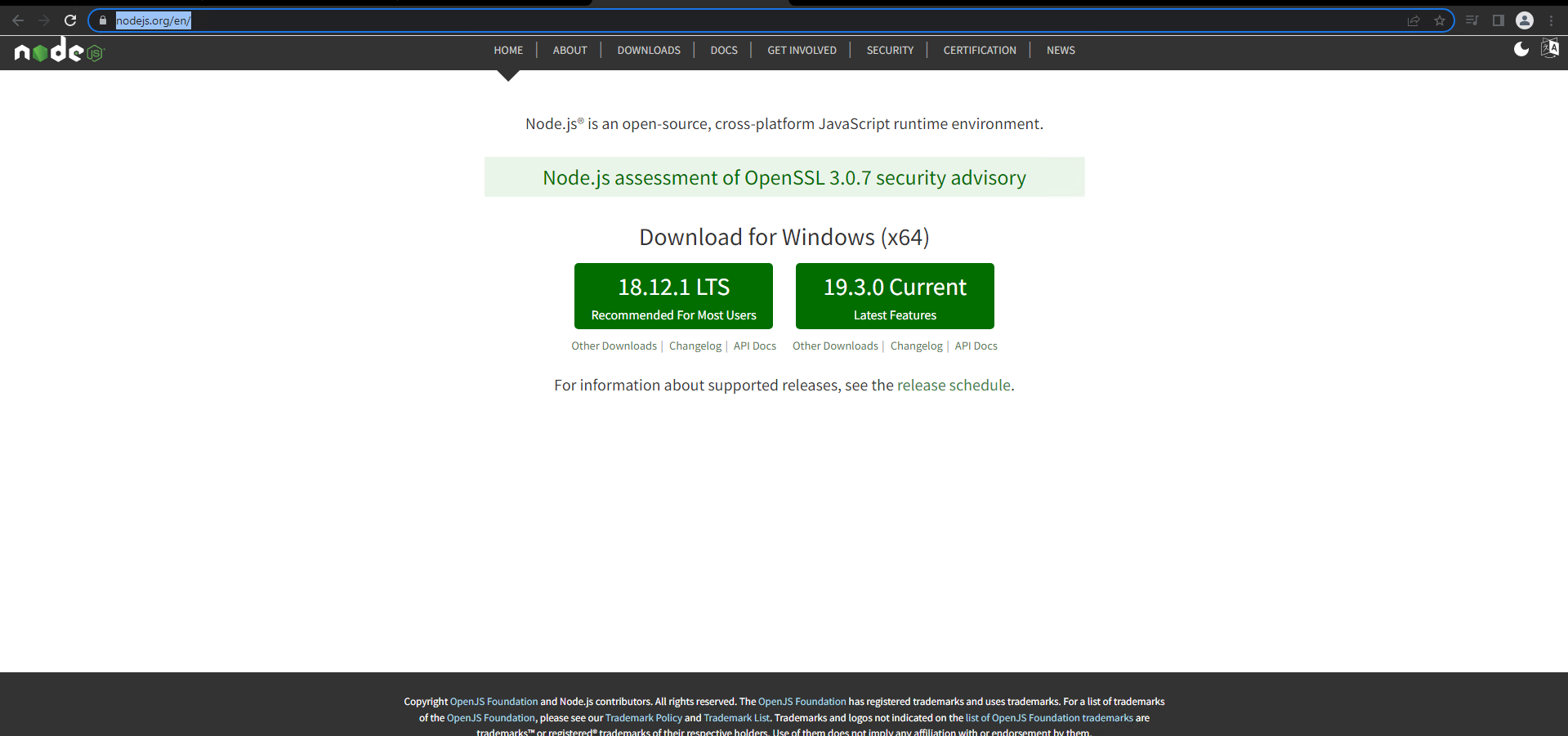
npm init –y

in cmd. If npm is not installed in your system. We should install npm.

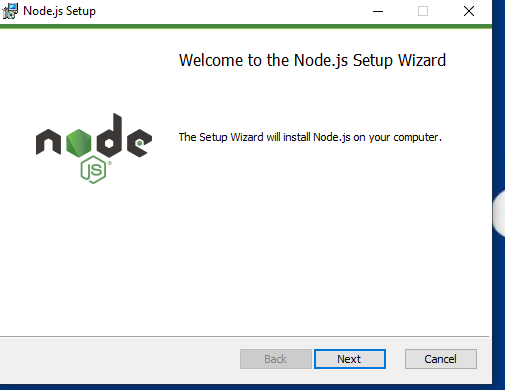
**Steps for installing npm**

Before installing npm we require **node.js**

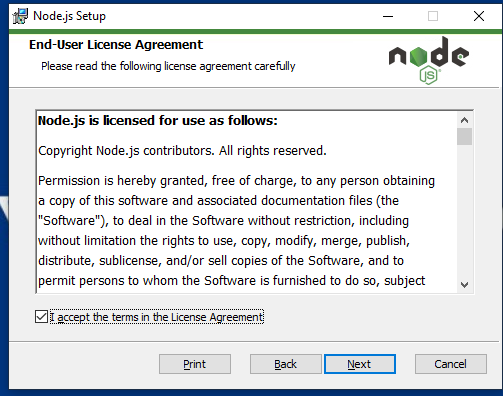
1. First go to this link <https://nodejs.org/en/>
2. Click the 18.12.1 version for node js

****

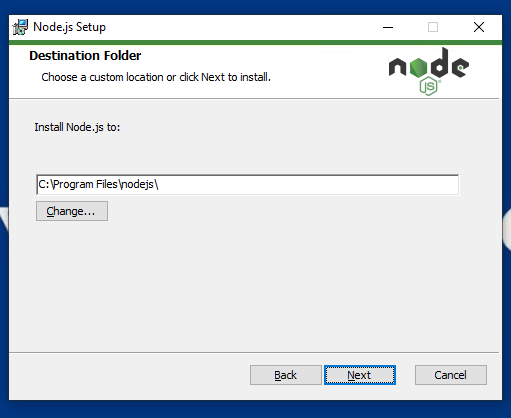
1. Double click the downloaded file and click the next button



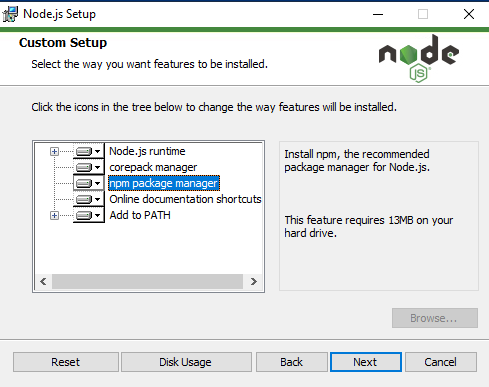
1. Tick the check box and click next button

****

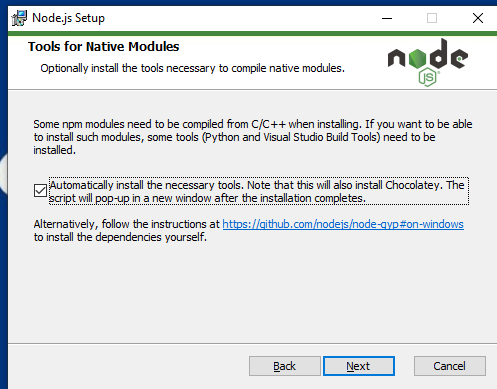
1. Click the next button



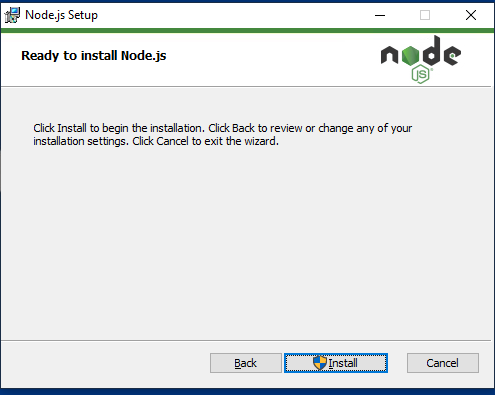
1. Select the npm package manager and Click the next button



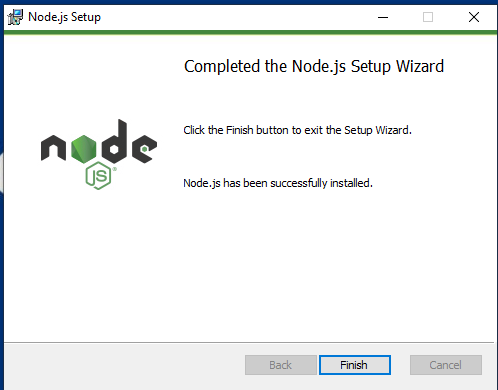
1. Tick the checkbox and click the next button



1. Click the install button



1. Click the finish button



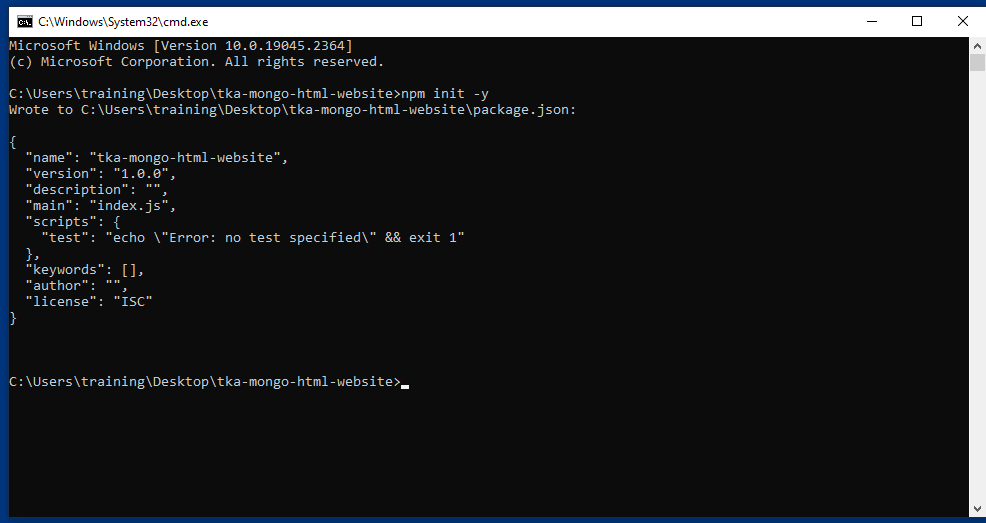
Node js is installed in our system. It will automatically install **npm** in our system.

Type ‘node –v’ (for checking nodejs version) in cmd. We can check the nod js is installed or not.

Type ‘npm –v’ (for checking npm version) in cmd. We can check the npm is installed or not

Next type the command in cmd

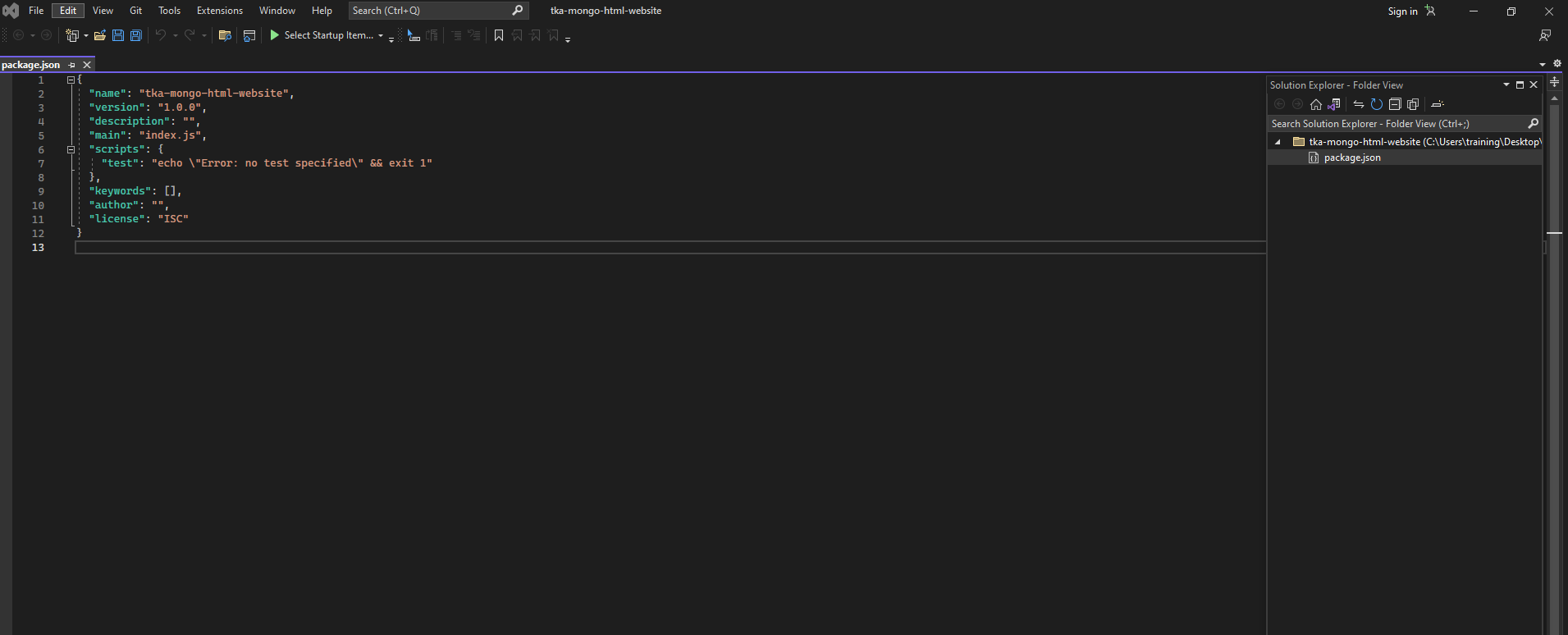
**npm init –y**

****

The command **npm init** creates a file package. json.

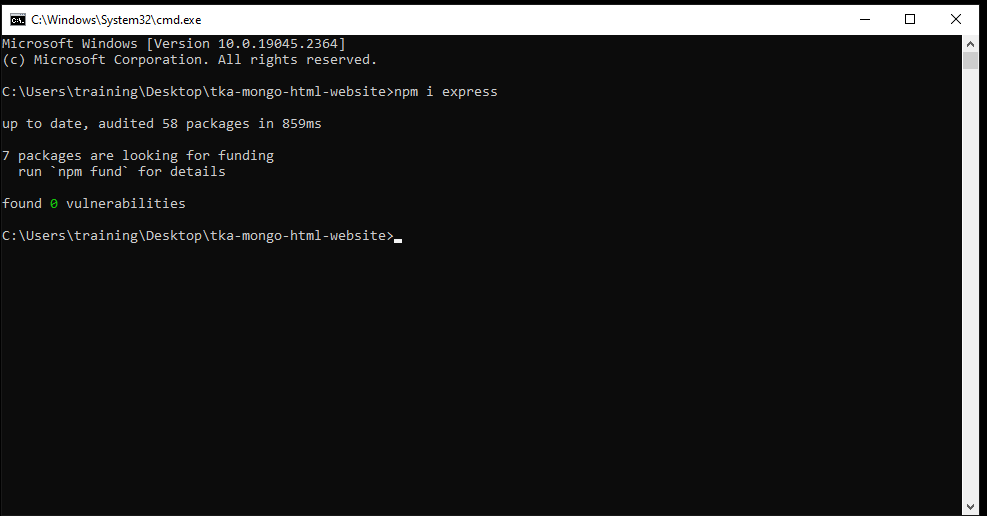
You can use it whenever you need to create a new Node.js project. When the file exists, you can add a dependency or devDependency to the file and install it using npm.

It will create a package. json in our project.

****

1. Next step is to type this command in npm

**npm i express**

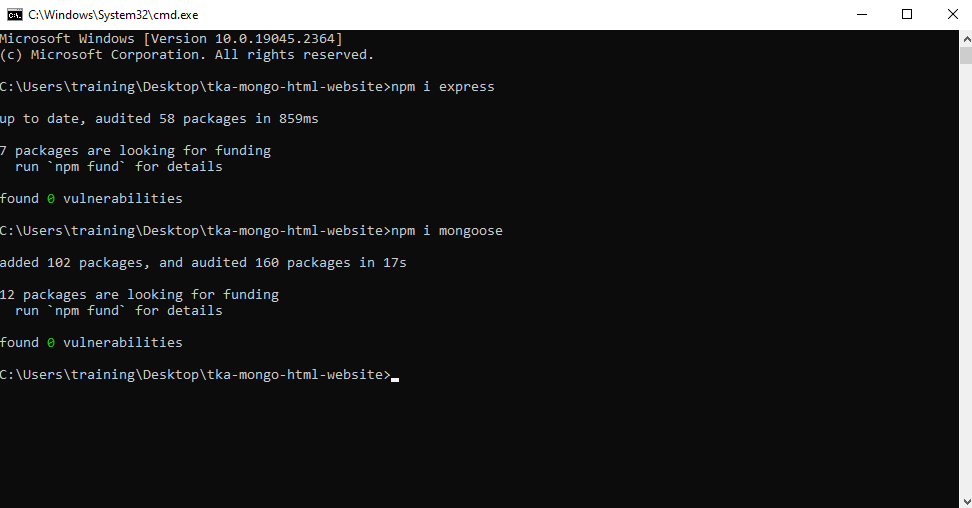


What is npm Express?

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

1. Install mongoose

npm i mongoose

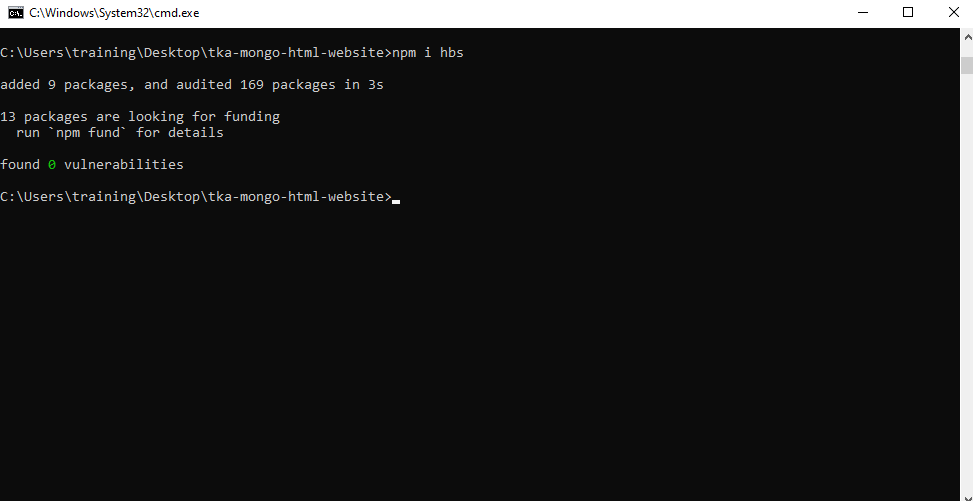


What is mongoose?

Mongoose **acts as a front end to MongoDB**, an open source NoSQL database that uses a document-oriented data model. A "collection" of "documents" in a MongoDB database is analogous to a "table" of "rows" in a relational database.

1. Next install hbs

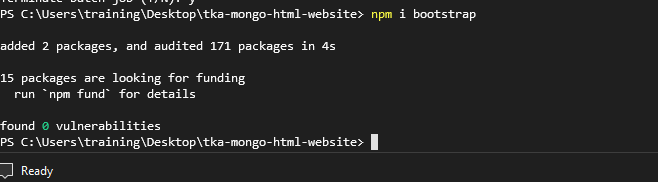
npm i hbs

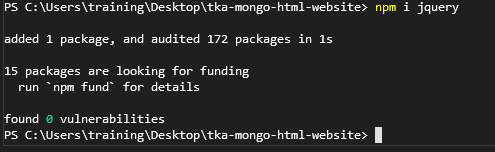
****

What is hbs?

hbs is **an express.** **js wrapper for the handlebars.** **js JavaScript template engine**. Handlebars. js is a template engine to make writing html code easier.

1. Next bootstrap and jquery should include in our project





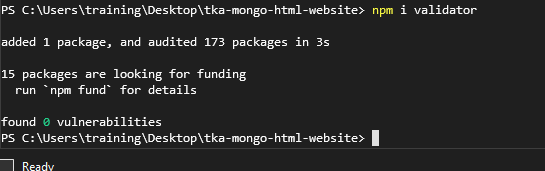
What is bootstrap?

Bootstrap is **a free, open source front-end development framework for the creation of websites and web apps**. Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs.

What is jquery?

The purpose of jQuery is **to make it much easier to use JavaScript on your website**. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

1. Install validator



What is validator?

A validator is a computer program used **to check the validity or syntactical correctness of a fragment of code or document**. The term is commonly used in the context of validating HTML, CSS, and XML documents like RSS feeds, though it can be used for any defined format or language.

Open our project folder in visual studio. Create 3 folder inside our project

**public**

**src**

**template**

In **public** folder we can keep the images and css.so create 2 folders inside of this.one is **css** and another one is **images**.

Inside of **src** folder create 2 folders, **db**, **app.js** and **model**.

**app.js**

const express = require("express");

const path=require("path");

require("./db/connection");

const User=require("./models/contactus");

const hbs=require("hbs");

const {registerPartials}=require("hbs");

const app =express ();

const port = process.env.PORT || 3000;

//setting the path

const staticpath=path.join(\_\_dirname,"../public");

const templatepath=path.join(\_\_dirname,"../template/views");

const partialpath=path.join(\_\_dirname,"../template/partials");

//middleware

app.use('/css',express.static(path.join(\_\_dirname,"../node\_modules/bootstrap/dist/css")))

app.use('/js',express.static(path.join(\_\_dirname,"../node\_modules/bootstrap/dist/js")))

app.use('/jq',express.static(path.join(\_\_dirname,"../node\_modules/jquery/dist")));

app.use(express.urlencoded({extended:false}))

app.use(express.static(staticpath));

app.set("view engine","hbs");

app.set("views",templatepath);

hbs.registerPartials(partialpath);

//routing

//app.get(path,callback)

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

//res.send("Hi I am awesome");

res.render("index");

})

app.post("/contact",async(req,res)=>{

try {

//res. send (req. body);

const userData =new User (req. body);

await userData.save();

res.status(201).render("index");

}

catch(error)

{

res.status(500).send(error);

}

})

//server create

app.listen(port,()=>{

console.log(`server is running at port number ${port}`);

})

* What does require Express mean?

var express = require('express'); => **Requires the Express module just as you require other modules and and puts it in a variable.**

Calls the express function "express ()" and puts new Express application inside the app variable.

* RegisterPartials provides a quick way to load all partials from a specific directory.
* What is process env port || 3000?

PORT || 3000 means: process. env. PORT means **the PORT number you manually set**. 3000 is the default port. If you haven’t set it manually then it will listen to 3000.

Inside **db folder** create a file **connection.js**

**conncetion.js**

const mongoose=require("mongoose");

mongoose.set ('strictQuery’, trues);

//creating a database

mongoose. connect ("mongodb://127.0.0.1:27017/tka-mongo-db",

{useNewUrlParser: true,

useNewUrlParser: true,

useUnifiedTopology:true

}

). then (()=>{

console.log ("connection successfull");

}). catch((error)=> {

console.log(error);

});

* What is require in Mongoose?

The require('mongoose') call above **returns a Singleton object**. It means that the first time you call require('mongoose'), it is creating an instance of the Mongoose class and returning it

* What is strict in Mongoose?

Mongoose is an ODM, and the {strict: true} option in this regard only applies to queries you run through Mongoose, it's not enforced in the mongoDB side, your documents remain schemaless.

* Mongoose.connect () is used to connect our database.

In **template folder** create 2 folders. **partials and views**.

* Partials:  **Handlebars** allows for template reuse through **partials**. **Partials** are normal **Handlebars** templates that may be called directly by others.
* Views contains **index.hbs** file

**Index.hbs**

<! DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

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

<link rel="stylesheet" href="../css/bootstrap.min.css">

<link rel="stylesheet" href="./css/style.css">

<title>tka-mongo-html website</title>

</head>

<body>

{{>navbar}}

<div class="container-fluid mt-5">

<div class="row">

<div class="col-md-10 col-12 mx-auto">

<div class="row">

<div class="col-md-6 col-12 hero-text order-md-0 order-1">

<h1>Training the world's professionals</h1>

<p>Expert training in a classroom, online or from home!</p>

<button class="btn1"><a href="https://www.theknowledgeacademy.com/" class="read">Read more</a></button>

</div>

{{!--right side --}}

<div class="col-md-6 col-12 hero-text order-md-1 order-0">

<figure>

<img src="images/online-class-pandemic.jpg" alt="">

</figure>

</div>

</div>

</div>

</div>

</div>

{{>contact}}

{{>about}}

<div id="service">

<div class="container-fluid mt-5">

<div class="row">

<div class="col-md-10 col-12 mx-auto">

<div class="row">

<div class="col-md-6 col-12 hero-text order-md-0 order-1">

<h1>Our Servicess</h1>

<p>Transforming organisations and individuals across the world. </p>

<p>Or select from our popular topics</p>

<h3>PRINCE2</h3>

<h3>SCRUM</h3>

<h3>AGILE</h3>

<h3>ITIL</h3>

<h3>PMP</h3>

</div>

{{!--right side --}}

<div class="col-md-6 col-12 hero-text order-md-1 order-0">

<figure>

<img src="images/service.jpg" alt="" class="serviceimg" width="400px;">

</figure>

</div>

</div>

</div>

</div>

</div>

</div>

<script src="../jq/jquery.js" type="text/java script"></script>

<script src="../js/bootstrap.js" type="text/Javascript"></script>

</body>

</html>

**Index.hbs** is the main file in this website.

**{{>navbar}}:** It will call the navbar.hbs file in partials.

* In partials contains the following files

Navbar.hbs,service.hbs, contact.hbs and about.hbs.

**contact.hbs**

<section class="my-4" id="">

<div class="py-4">

<h2 class="text-center">Contact</h2>

</div>

<div class="w-50 m-auto">

<form action="/contact" method="post">

<div class="form-group">

<label>Name:</label>

<input type="text" name="name" class="form-control"> </div>

<div class="form-group">

<label>Email:</label>

<input type="text" name="email" class="form-control"> </div>

<div class="form-group">

<label>Phone Number:</label>

<input type="text" name="phone" class="form-control">

<label>Message:</label>

<input type="textarea" name="message" class="form-control"> </div>

<div class="form-group">

</div>

<div class="button">

<button type="submit" class="btn btn-success">Submit</button></div>

</form>

</section>

When we press the submit button the details will be saved to mongodb.

* First form action goes to the route ‘/contact’. In app.js we can see the following line.

const User=require("./models/contactus");

Here User will get the contactus model details. That is in our model we added name, email,phosne and message.

**contactus.js**

const mongoose=require("mongoose");

const validator=require("validator");

const userSchema =monsgoose.Schema({

name:{

type:String,

required:true,

minLength:3

},

email:{

type:String,

required:true,

validate(value){

if (! validator.isEmail(value)){

throw new Error("Invalid email id")

}

}

},

phone:{

type:Number,

required:true,

min:10

},

message:{

type:String,

required:true,

minLength:3

}

})

//we need a collection

const User=mongoose.model("User”, userSchema);

module. Exports=User;

* Our collection name is users. So our model name should be singular name. That is User.

**app.js**

app.post("/contact",async(req,res)=>{

try {

//res. send (req. body);

const userData =new User (req. body);

await userData.save();

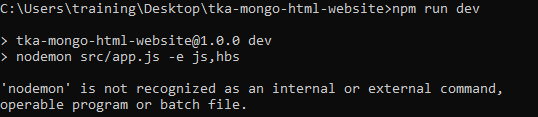
res. status (201). render("index");

}

* This will get data from contact and save to Database collection name named users.
* We can run the website by typing

**npm run dev** in cmd

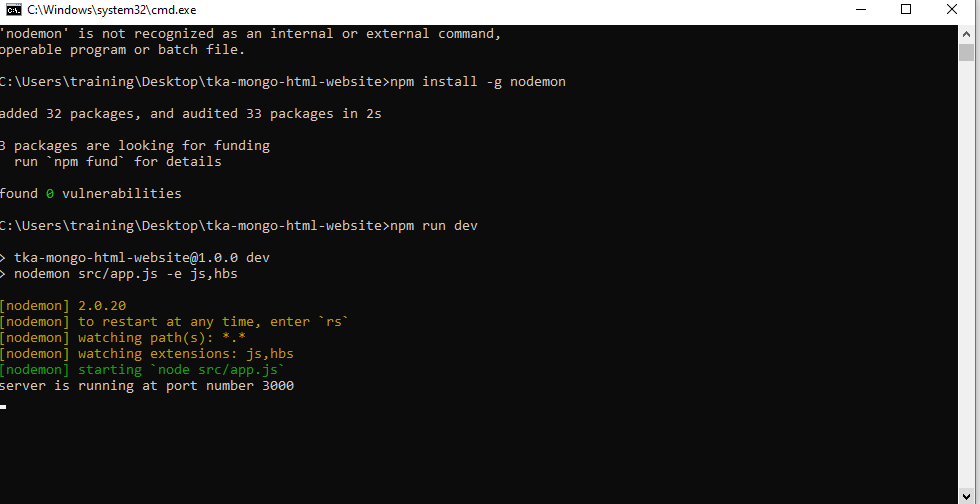
Sometime run npm will get the following error.



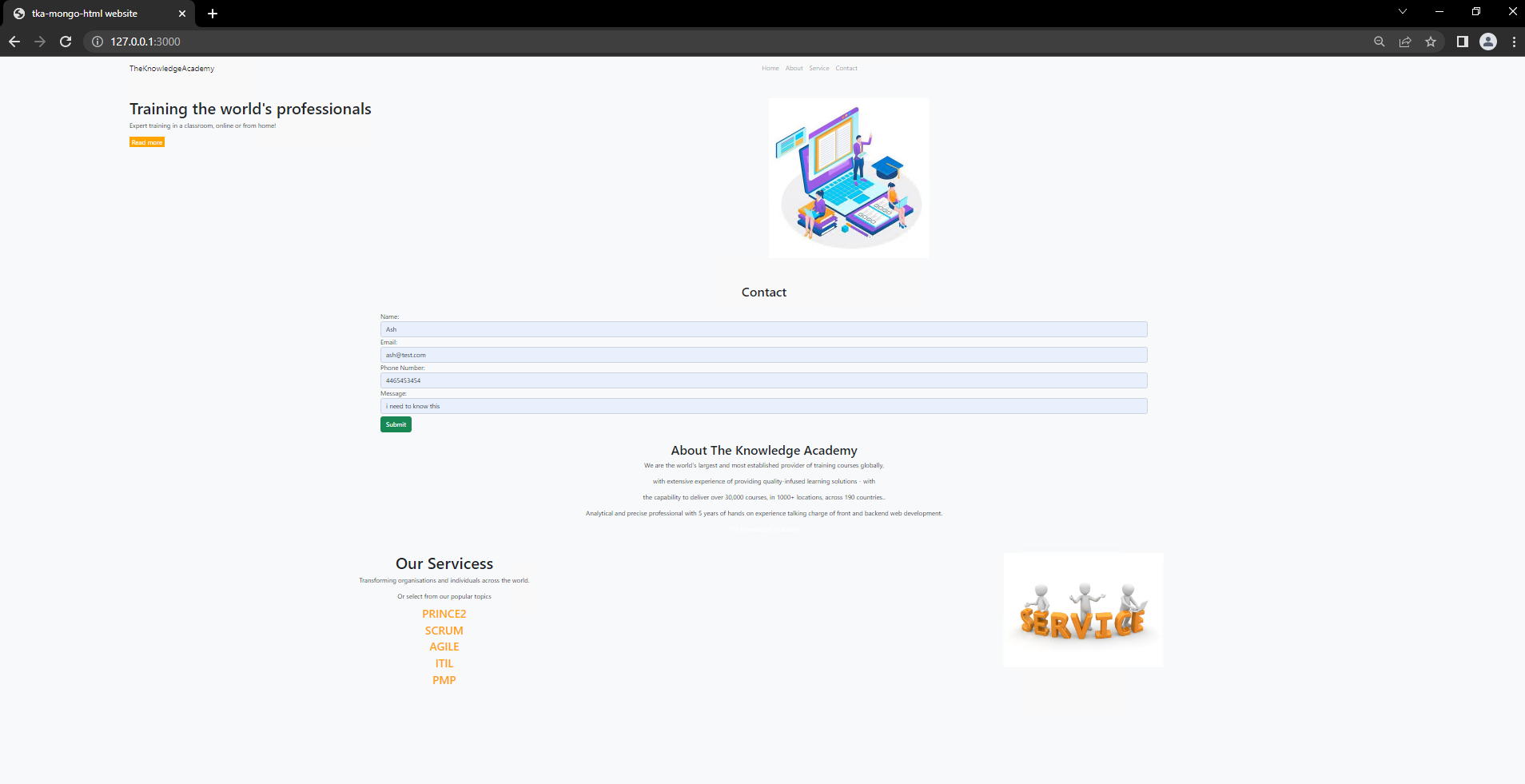
What is nodemon?

Nodemon is **a command-line tool that helps with the speedy development of Node.** **js applications**. It monitors your project directory and automatically restarts your node application when it detects any changes. This means that you do not have to stop and restart your applications in order for your changes to take effect.

If nodemon is not installed in our system. Install nodemon. Then run npm.



Our final Website is as follows:



Following shows the contact form.



* Our resultant database will be as following.

