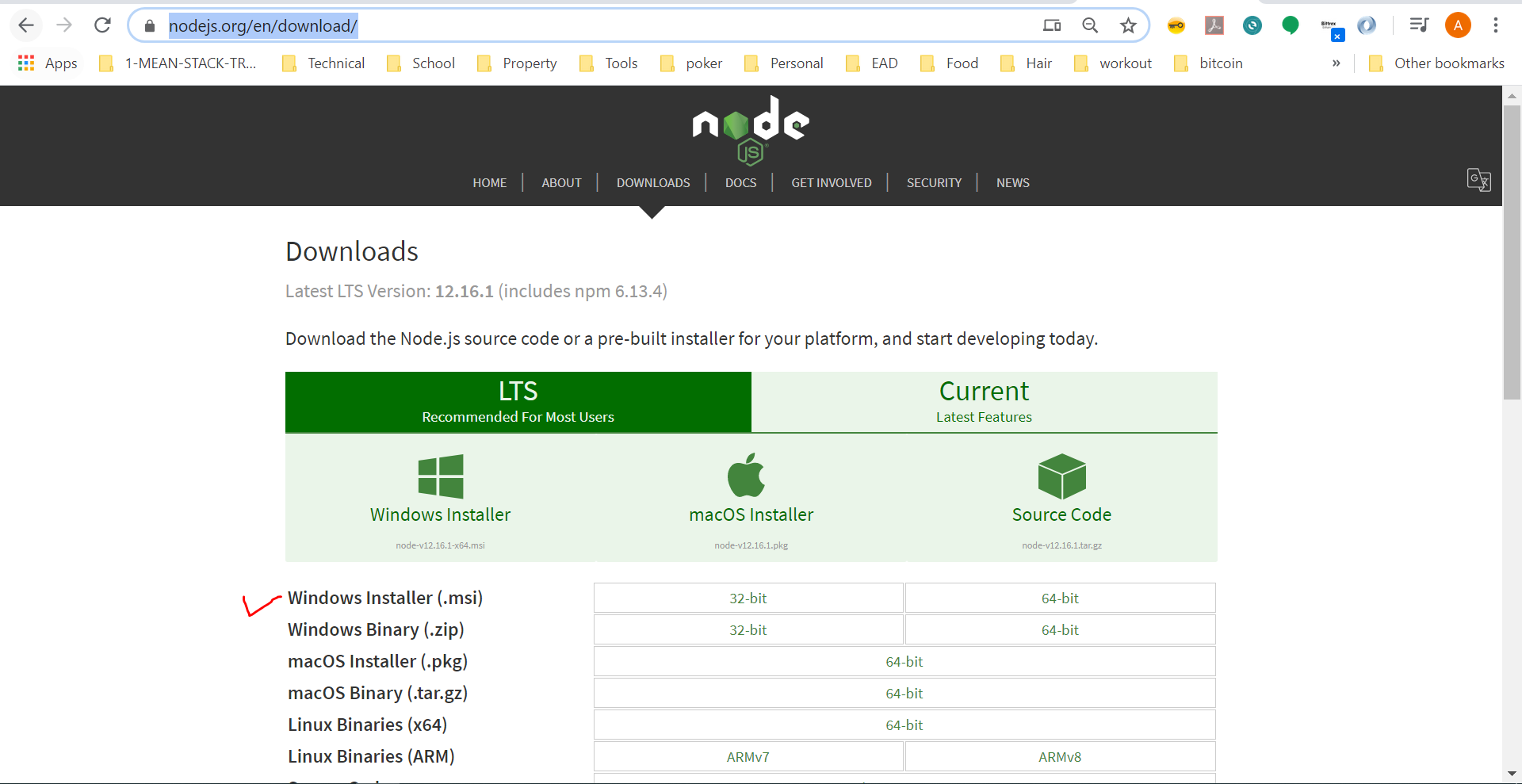
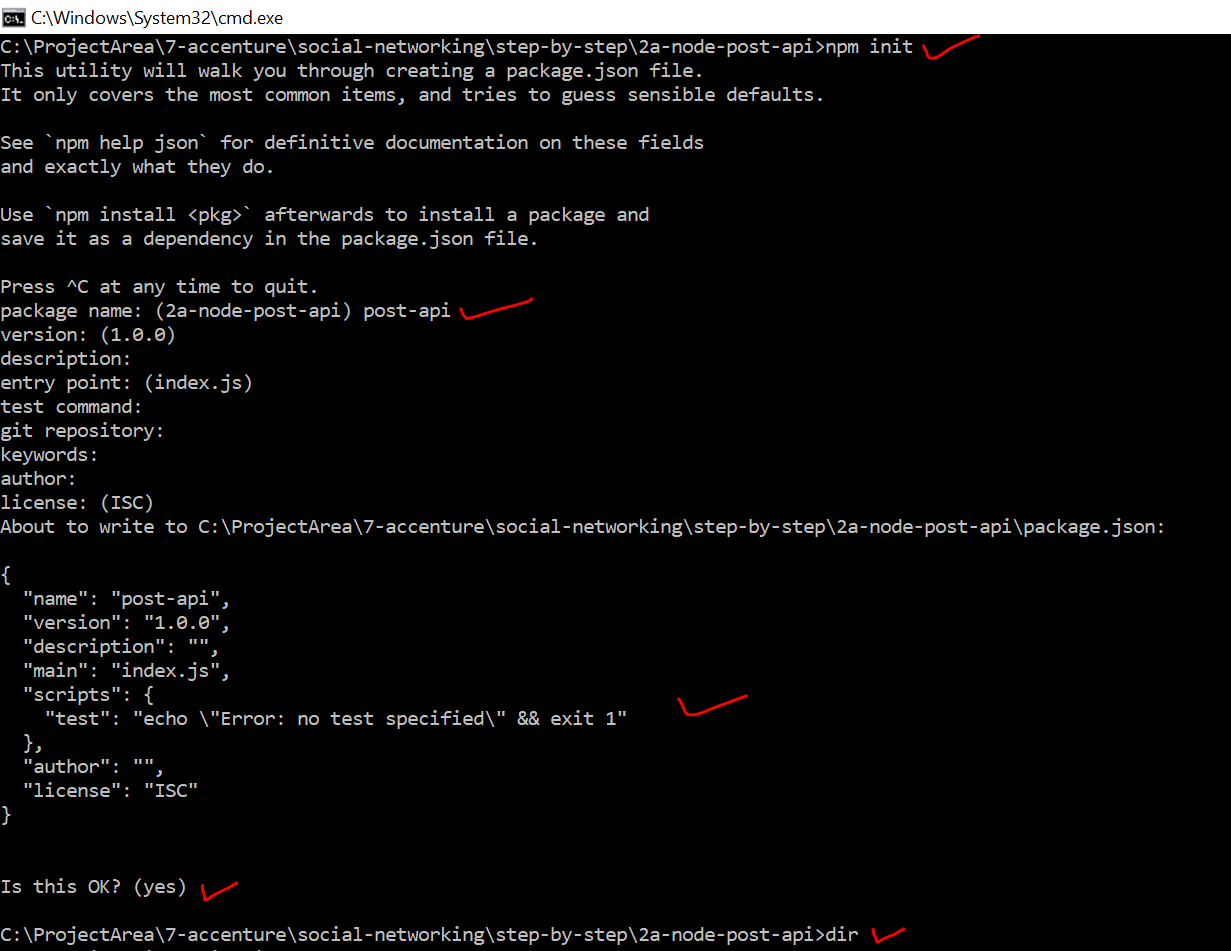
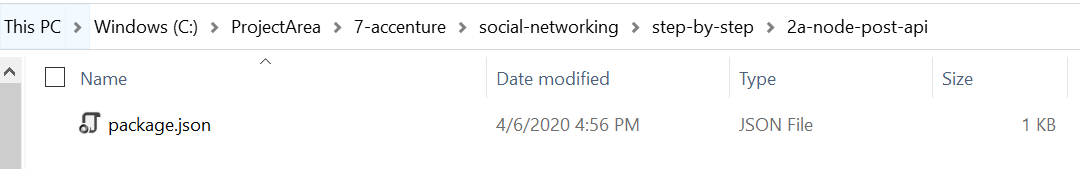
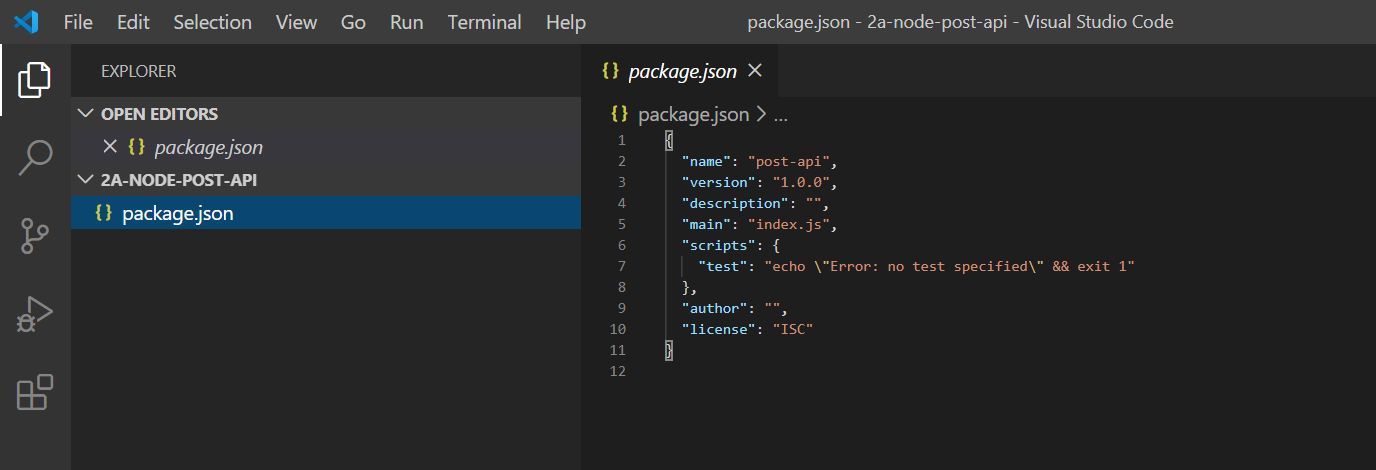
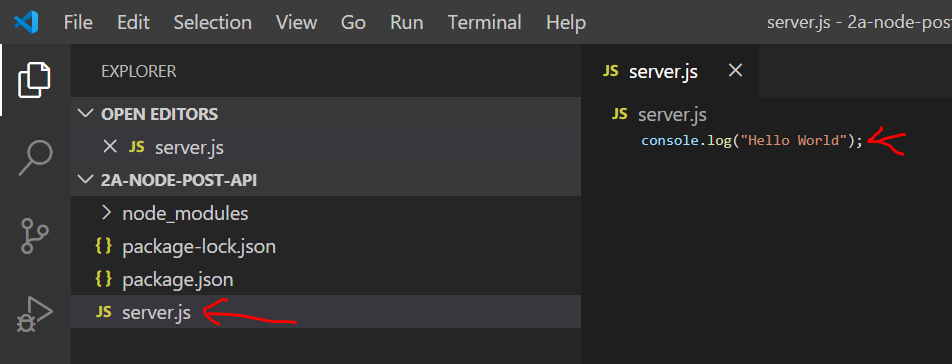
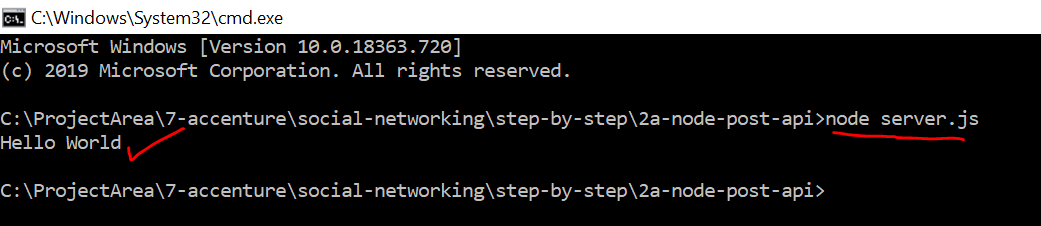
Installation

1. Install node runtime from nodejs.org - <https://nodejs.org/en/download/>
   1. 
2. **Create a node project folder name “post-api”** 
   1. **Inside the folder execute a command “npm init” to start a new node project**
   2. 
3. After the above step we have single file called package.json created to begin with
   1. 
4. Open the project folder in visual studio code and observe the package.json
   1. 
5. As node can execute a JavaScript code , lets write a simple hello world javascript program and execute inside node runtime. Please note this is not a server yet
   1. 
   2. Run command -> node server.js
   3. Observe the output
   4. 
6. Let’s convert this to a server by importing http package
   1. Copy the below code to server.js file and execute command -> node server.js

// create a server variable

const http = require("http");

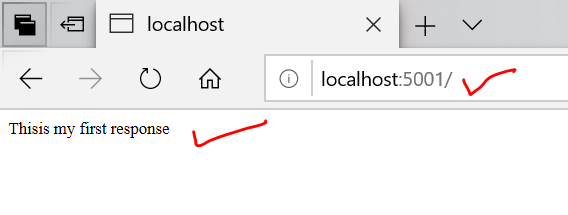
const server = http.createServer((req, res) => {

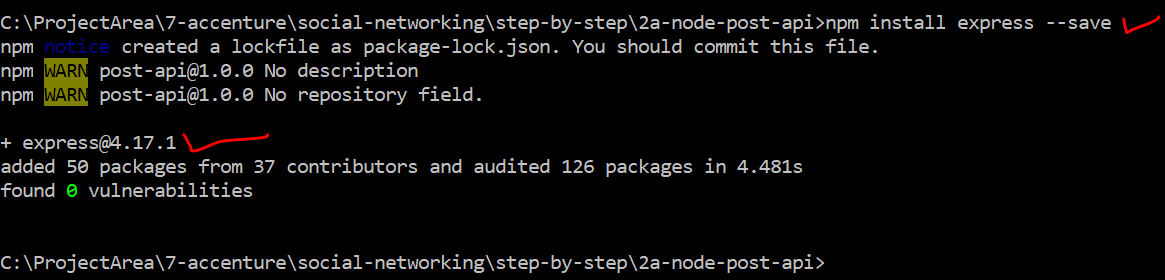
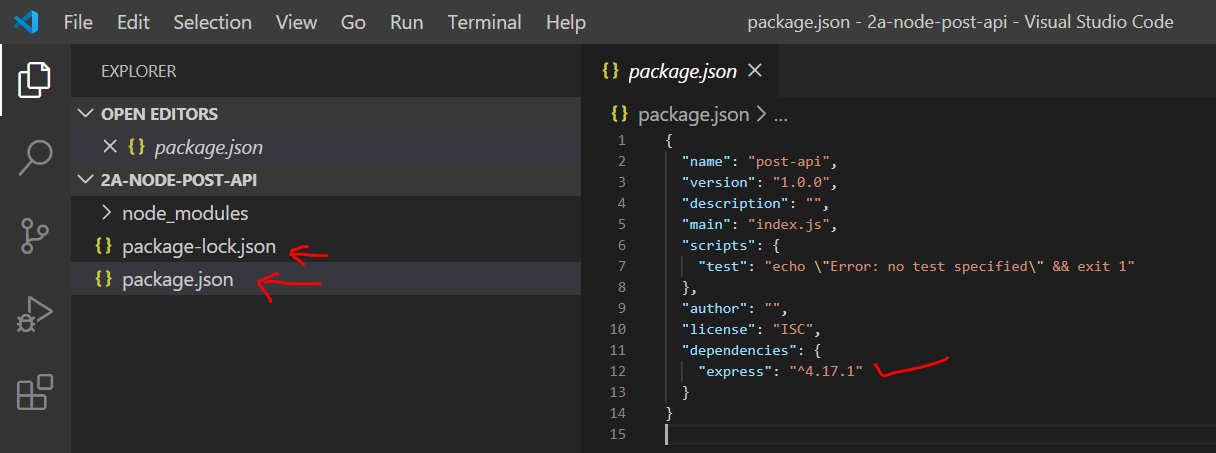
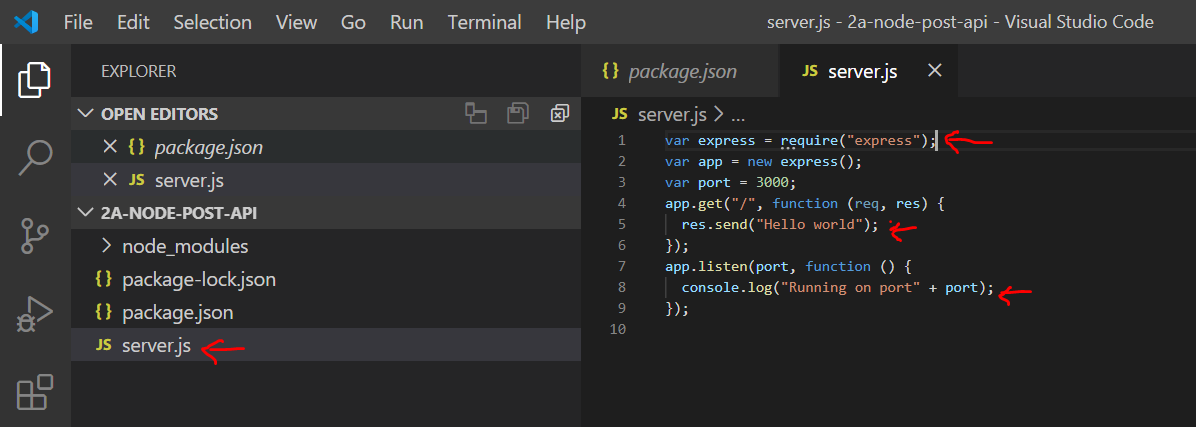
  res.end("This is my first response");

});

// listen at a port

server.listen(5001);

* 1. Observe the response in browser
     1. 

1. As server needs to listen to http request and understand GET, POST, PATCH, DELTE request verbs and parse request data, we need to write a lot of code in nodejs to achieve this
2. This is made easy by using another package called express
3. Let’s listen to the client request using express package
4. **Install Express by running command “npm install express --save”**
   1. 
5. **Open the code is editor and notice package.json , express dependency is installed**
   1. 
6. Simple ‘Hello World’ using express
   1. Make below changes to server.js and observer the highlighted areas
   2. 
   3. Include the below Code in server.js

var express = require("express");

var app = new express();

var port = 5000;

app.get("/", function (req, res) {

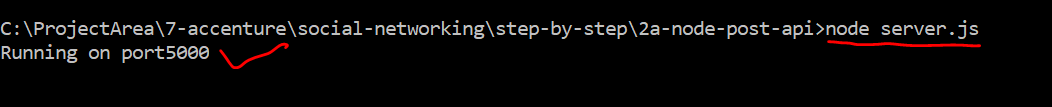
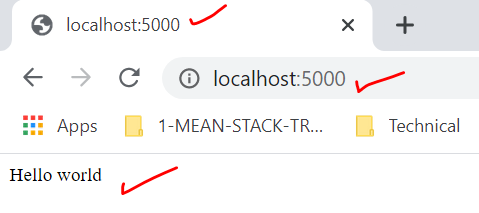
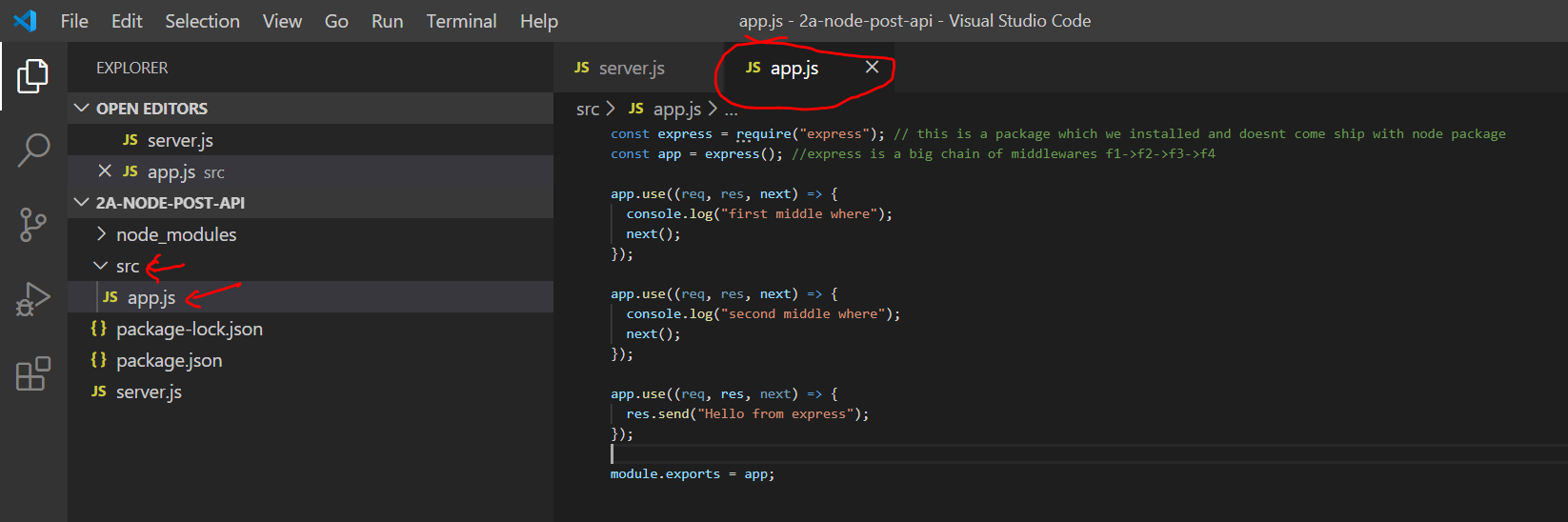
  res.send("Hello world");

});

app.listen(port, function () {

  console.log("Running on port" + port);

});

1. Run the server and check the output
   * 1. Run command -> “node server.js”
     2. 
2. Type localhost:5000 is chrome(browser) to get the message
   1. 
3. Middleware concept using express – express is nothing but a chain of middleware’s
   1. Create a new folder called src
   2. Create app.js file and include below code
   3. 
   4. Code – include in app.js

const express = require("express"); // this is a package which we installed and doesnt come ship with node package

const app = express(); //express is a big chain of middlewares f1->f2->f3->f4

app.use((req, res, next) => {

  console.log("first middle where");

  next();

});

app.use((req, res, next) => {

  console.log("second middle where");

  next();

});

app.use((req, res, next) => {

  res.send("Hello from express");

});

module.exports = app;

* 1. Include the above exported app object into server.js file with below code

const http = require("http");

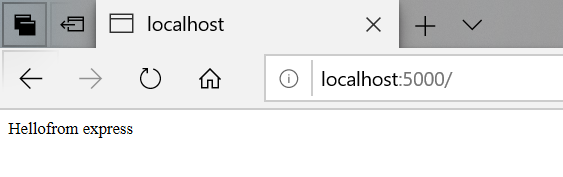
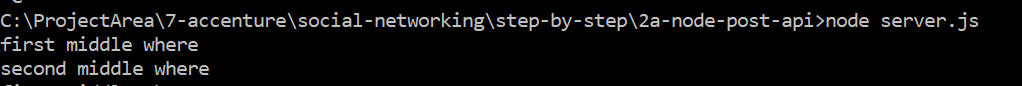
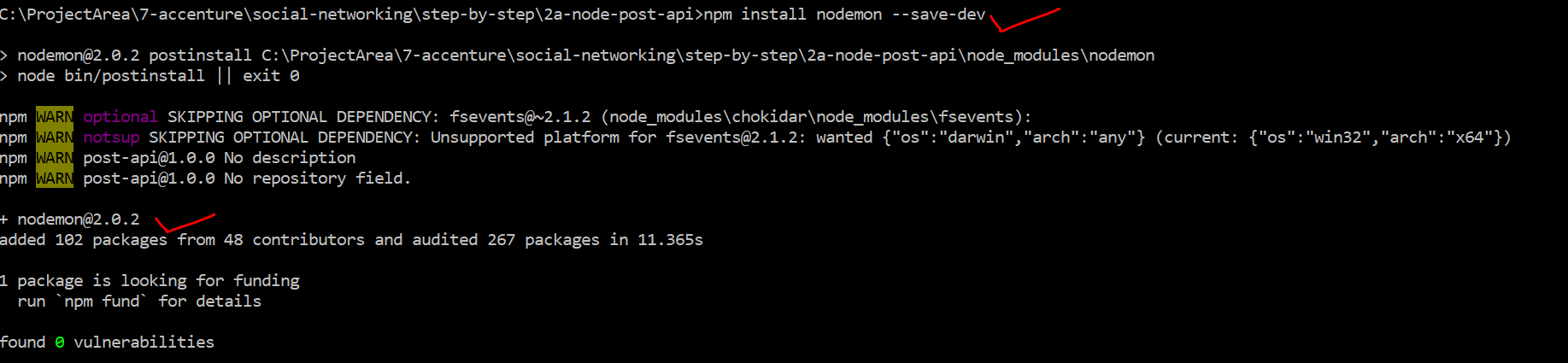
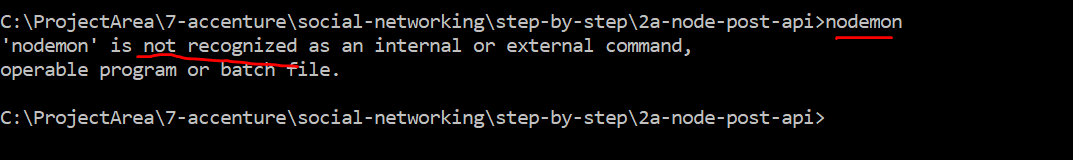
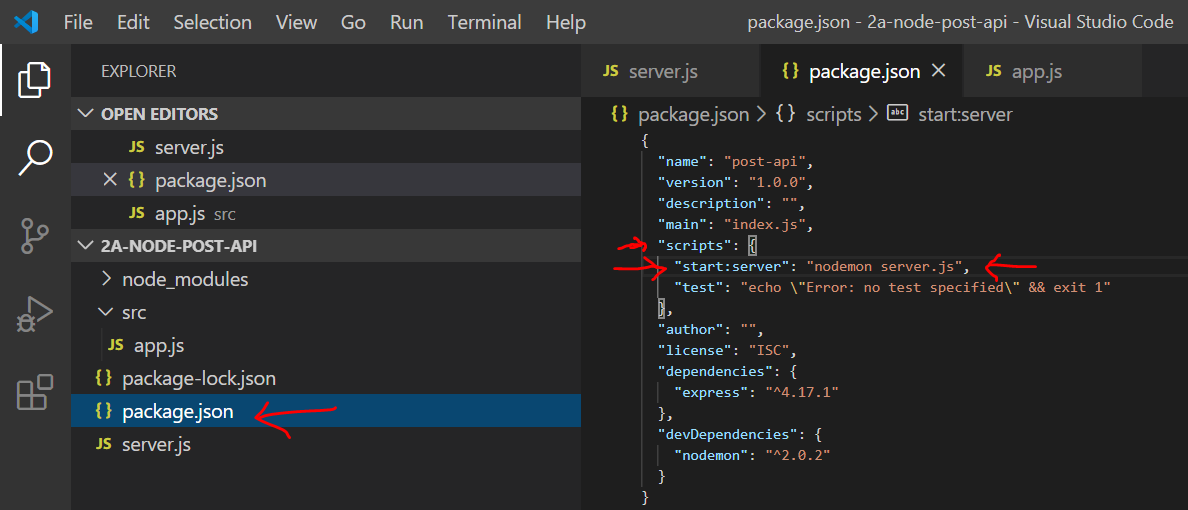
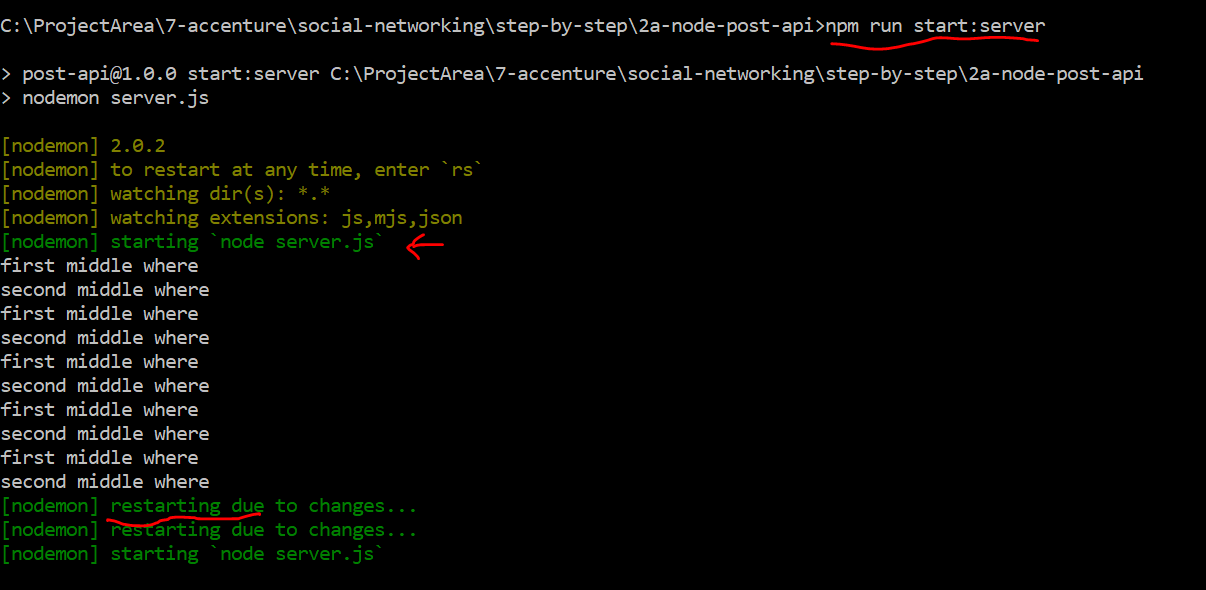
const app = require("./src/app");

const port = process.env.PORT || "5000";

app.set("port", port);

const server = http.createServer(app);

server.listen(port);

1. Execute the project and see the output in browser and console
   1. 
   2. 
2. Every time we are doing code changes, we are killing the server and restarting. If we have to make our server recompile and restart automatically whenever new code changes happen, we can do this with help of nodemon
   1. Run command -> npm install nodemon --save-dev
      1. 
   2. NOTE : nodemon is installed locally for this project only and it won’t run the command
      1. 
   3. In-order to run nodemon create a custom command as shown below in package.json under scripts section
      1. 
      2. Run the server and do some code changes and observe that server restarts automatically
         1. 
3. Create get api
   1. Paste the below code in app.js and run <http://localhost:5000/api/posts>

const express = require("express");

const app = express();

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

  const posts = [

    {

      id: "fadf12421l",

      title: "First server-side post",

      content: "This is coming from the server",

    },

    {

      id: "ksajflaj132",

      title: "Second server-side post",

      content: "This is coming from the server!",

    },

  ];

  res.status(200).json({

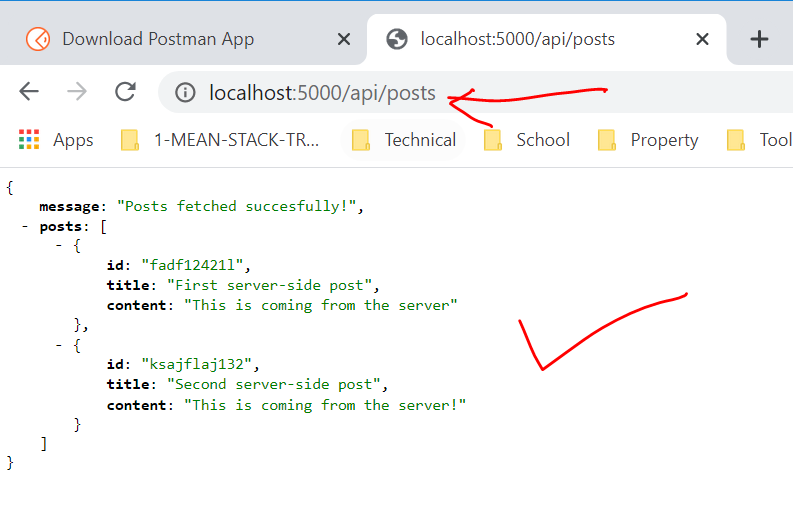
    message: "Posts fetched succesfully!",

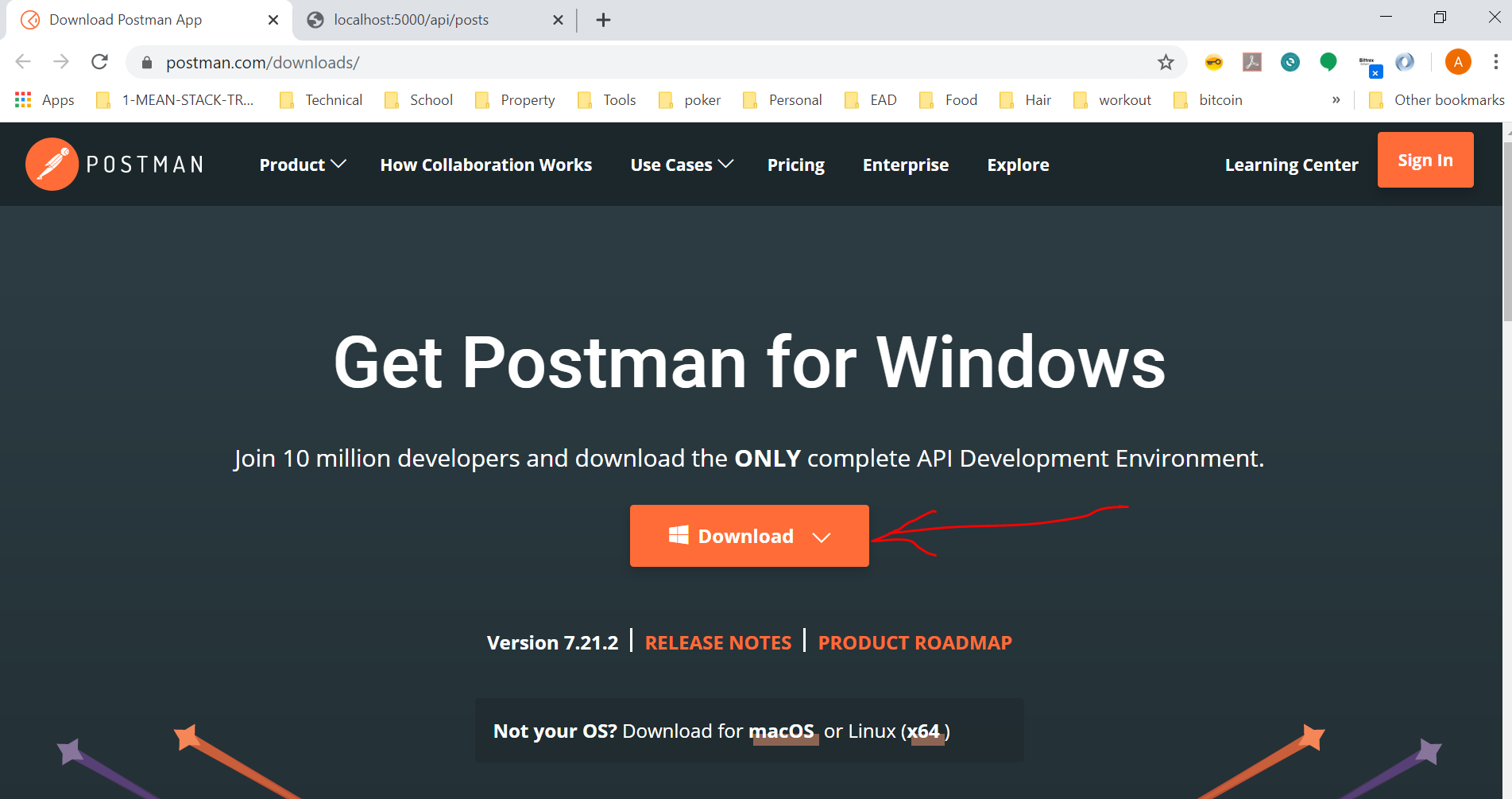
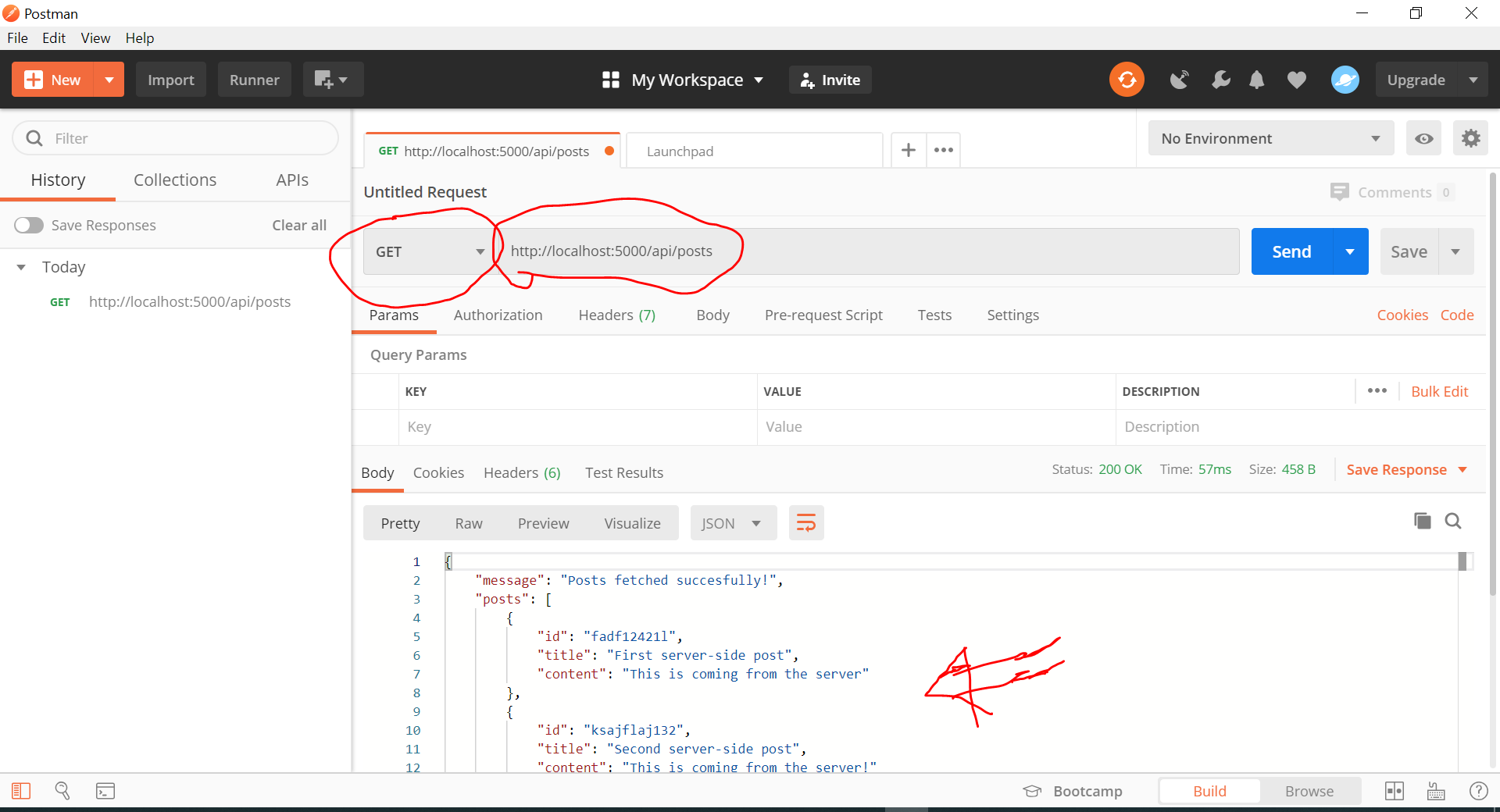
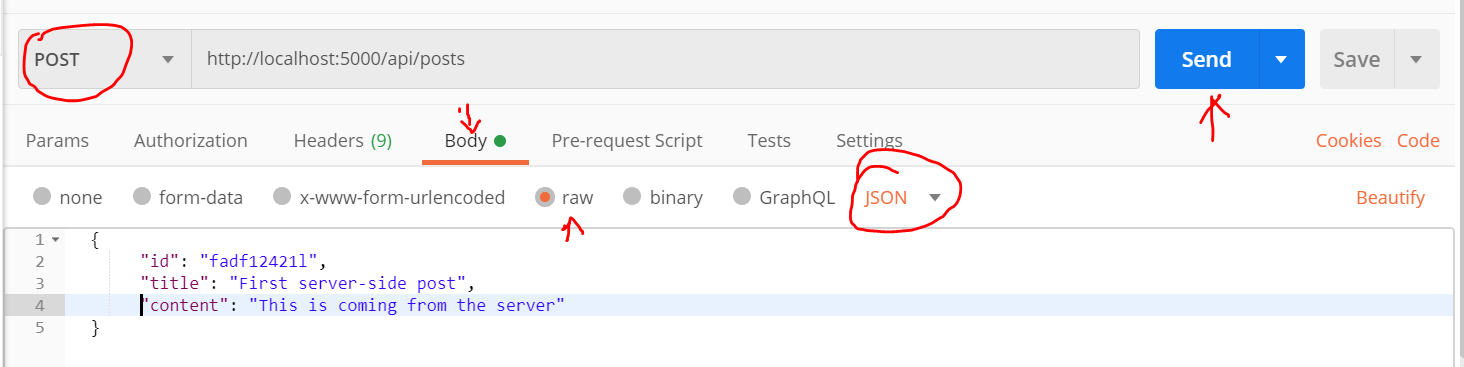
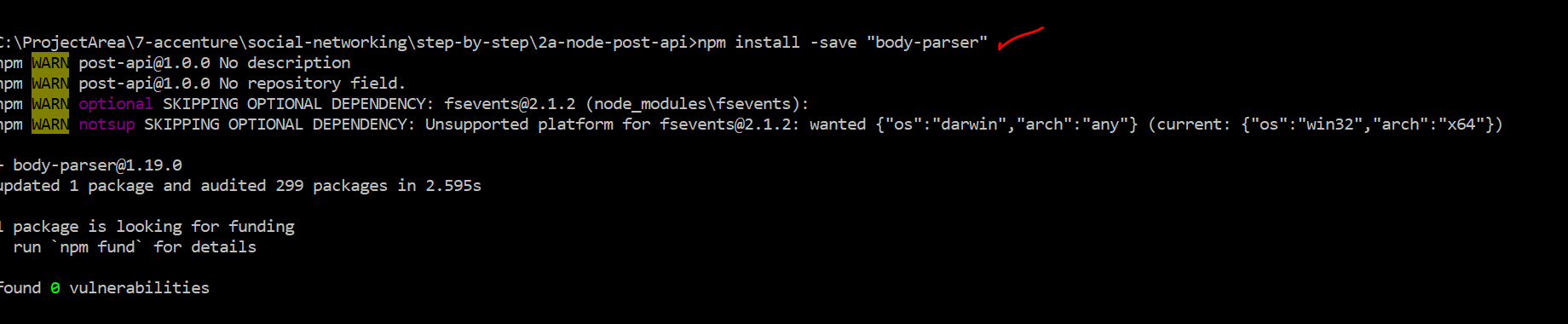
    posts: posts,

  });

});

module.exports = app;

* 1. Observer the output
  2. 

1. Install POSTMAN tool from below site- <https://www.postman.com/downloads/>
   1. Postman tool will be used perform all types http requests
   2. 
   3. For example: At this point we have GET API and that can be executed in POSTMAN as below
      1. 
2. Next step is to Create Post API – This API is responsible for creating out POST records
   1. NOTE: User can initiate only HTTP GET requests using browser, can not do POST or PUT or DELETE requests.
   2. Our requirement is to initiate a HTTP POST request using POSTMAN tool as shown below and get a response saying the record was created successfully
      1. 
   3. We have not created POST API in server yet
   4. POST API should accept data from POST http request body
   5. The POST http request body data need to be extracted from the request and parsed made available in request variable at the server side. This is performed by a middleware package called “body-parser”. Let’s install “body-parser” with below command
      1. npm install –save “body-parser
      2. 
      3. This is a node express package , which is used as middlerware.
   6. Lets start building the POST API with below code
      1. Import the package in app.js
         1. const bodyParser = require("body-parser");
      2. Lets attach this middleware into our request pipeline with below code

// Lets attach the body-parser middleware

// bodyParser.json() -> this will tell only to process json type data from the request body

app.use(bodyParser.json());

//another example showing body-parser can process other types of body other than json

app.use(bodyParser.urlencoded({ extended: false }));

* + 1. Add below post API code

app.post("/api/posts", (req, res, next) => {

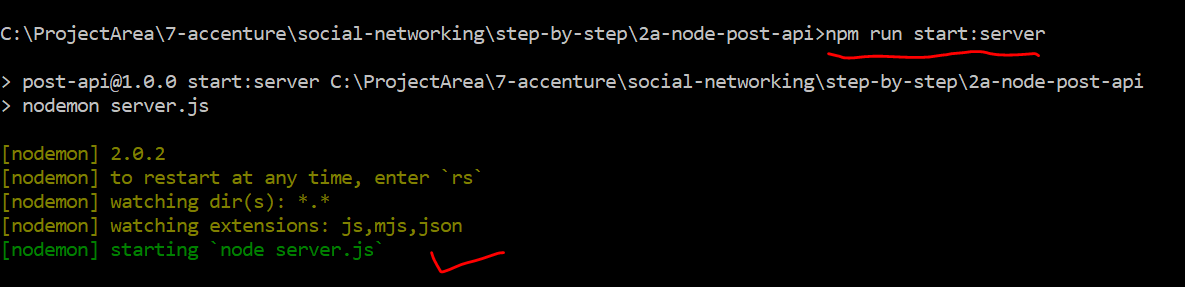
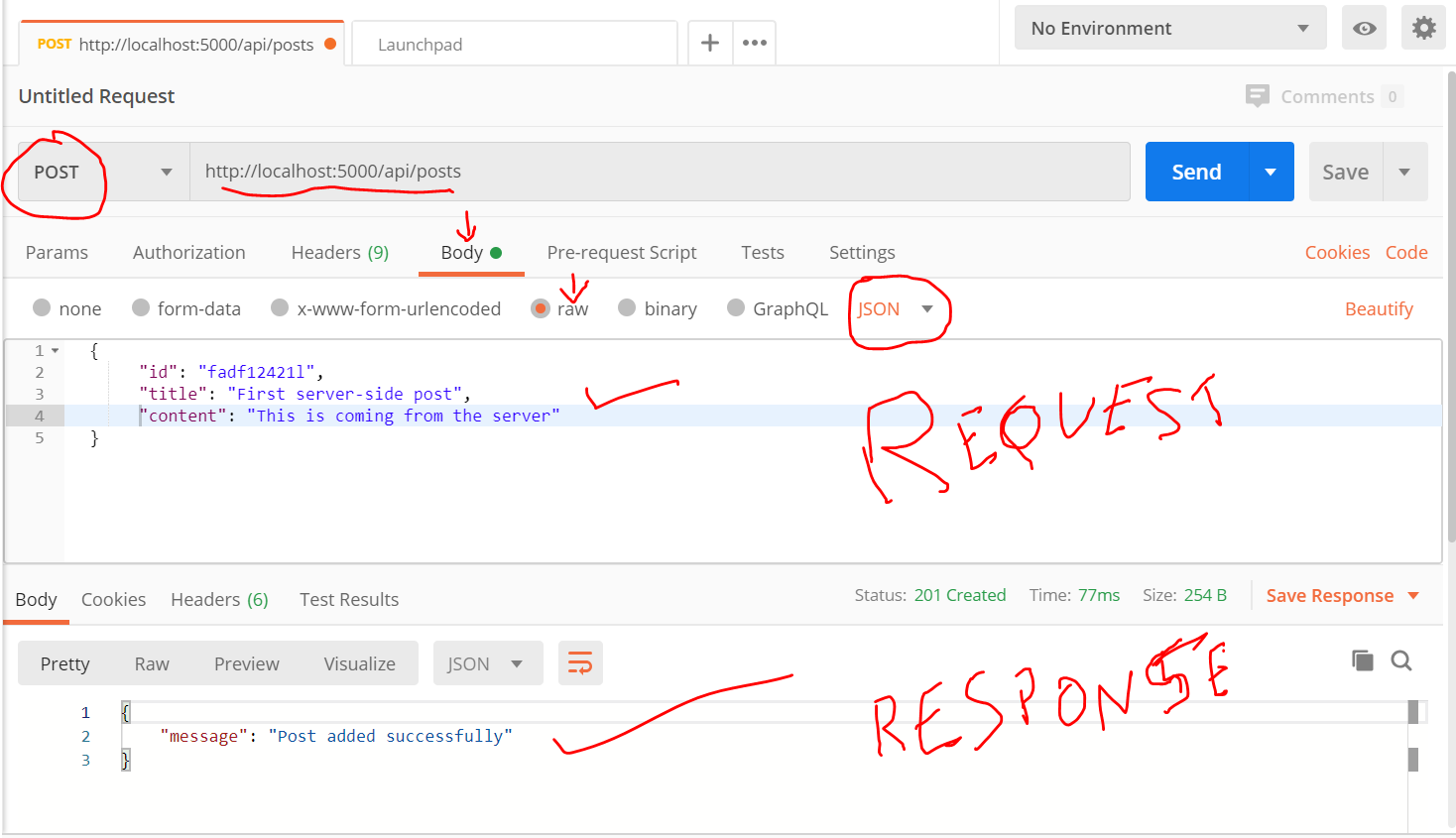
  const post = req.body;

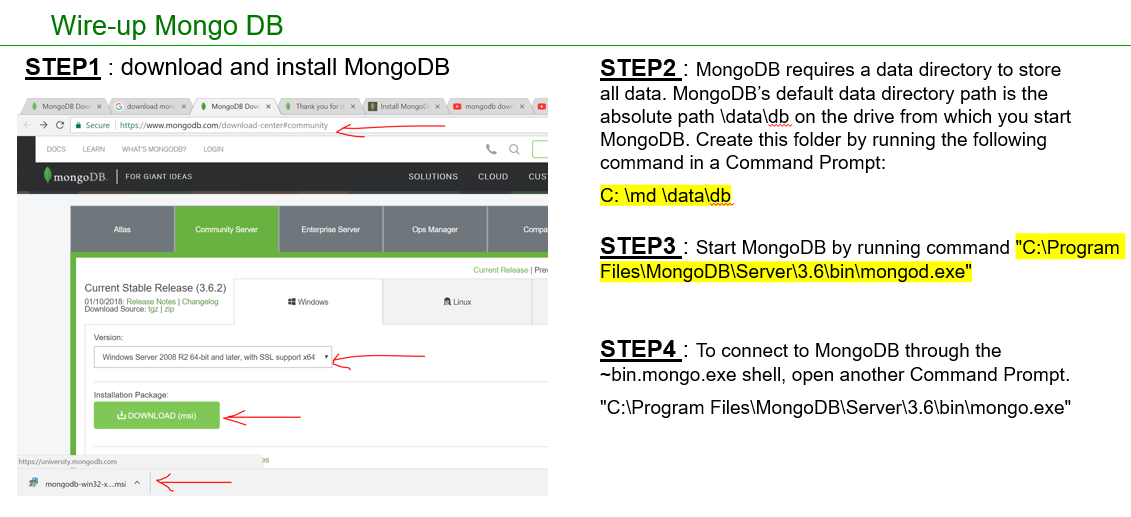
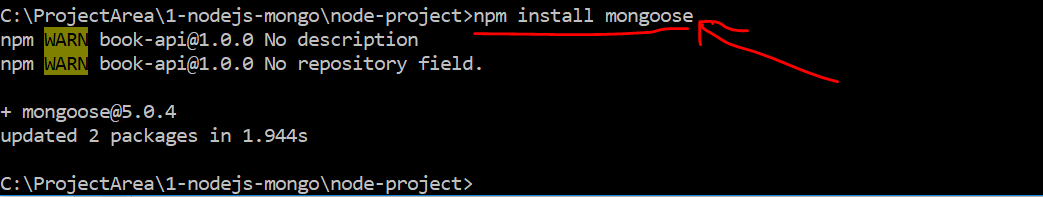
  res.status(201).json({

    message: "Post added successfully",

  });

});

* + 1. Start the server using command
       1. 
    2. Initiate the POST request in POSTMAN and observer the response
       1. 

1. Next step is to store the data from POST API into mongoDB
2. Wire up MongoDB
   1. 
3. Install mongoose using npm install mongoose
   1. 
4. Add a file bookModel.js and paste below code

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var bookModel = new Schema({

title : {

type : 'string'

},

author : {type : 'string'},

genere : {type : 'string'},

read : {type : Boolean, default: false}

});

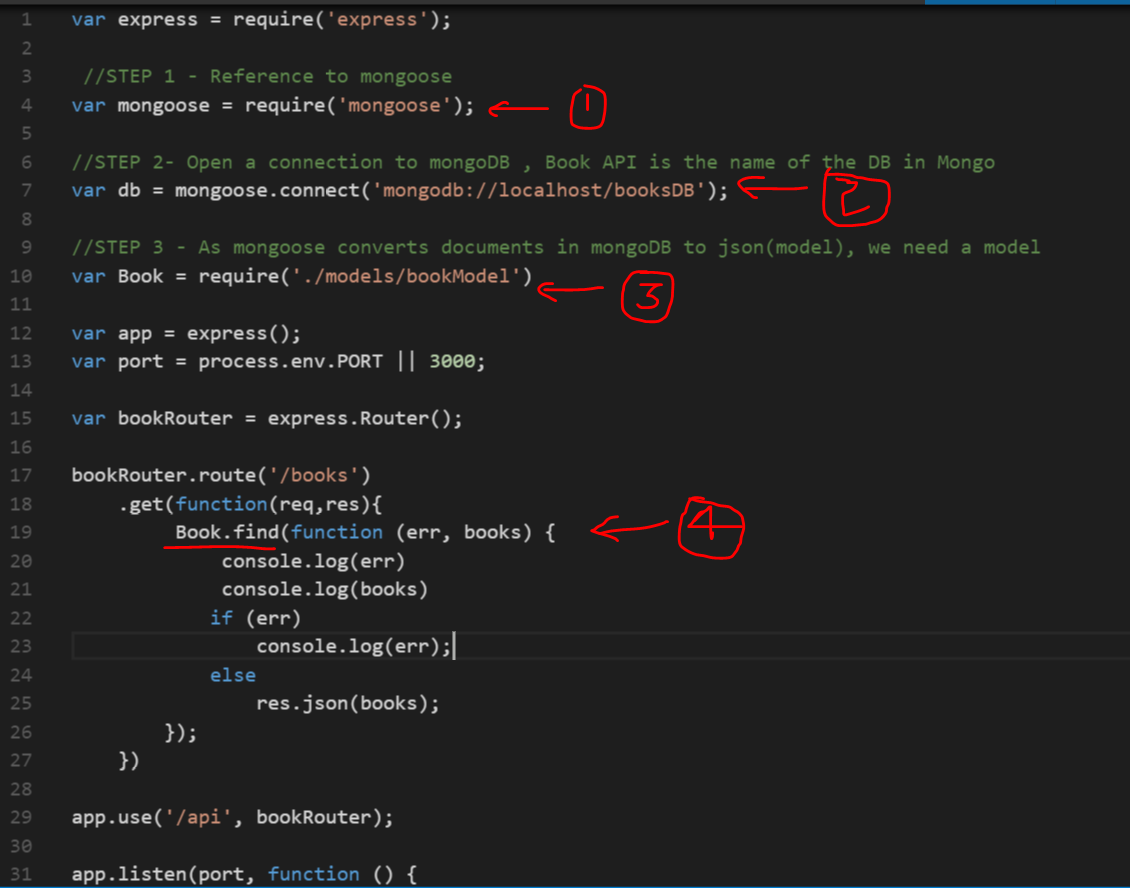
// This is going to add a new model called 'Book' into mongoose.Schema

// which later can be accessed in app.js using 'Book'

// Connectin string -> 'mongodb://localhost/booksDB'

// collection under booksDB is 'booksCollection' which is the third parameter

module.exports = mongoose.model('Book',bookModel,'booksCollection');

* 1. Update app.js with below code
  2. 

var express = require('express');

//STEP 1 - Reference to mongoose

var mongoose = require('mongoose');

//STEP 2- Open a connection to mongoDB , Book API is the name of the DB in Mongo

var db = mongoose.connect('mongodb://localhost/booksDB');

//STEP 3 - As mongoose converts documents in mongoDB to json(model), we need a model

var Book = require('./models/bookModel')

var app = express();

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

var bookRouter = express.Router();

bookRouter.route('/books')

.get(function(req,res){

Book.find(function (err, books) {

console.log(err)

console.log(books)

if (err)

console.log(err);

else

res.json(books);

});

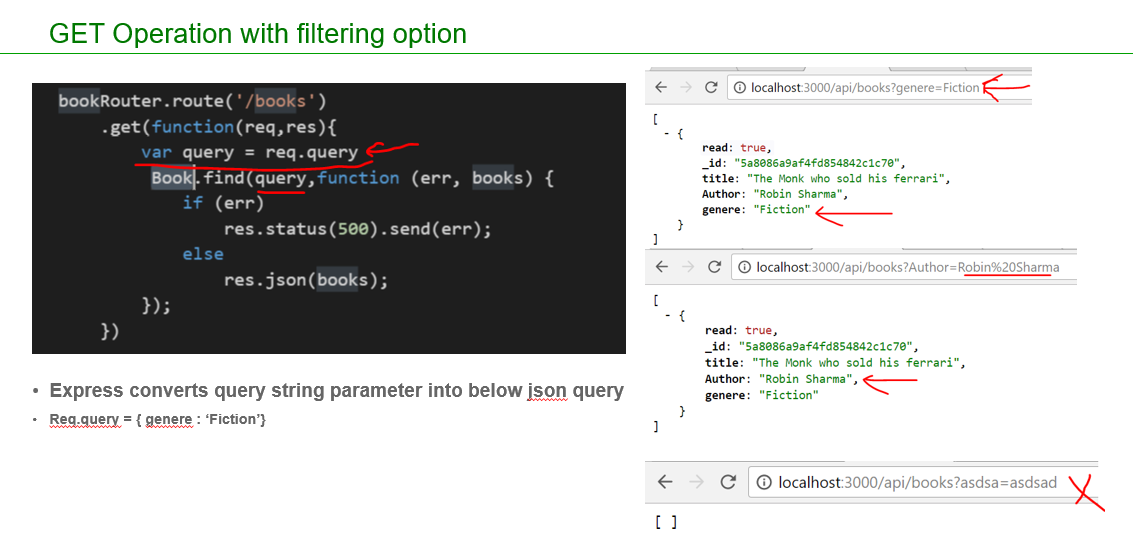
})

app.use('/api', bookRouter);

app.listen(port, function () {

console.log('Running on port ' + port);

})

1. GET API with filtering
   1. 
   2. Place the below code appropriately in app.js ( do not overwrite entire app.js)

bookRouter.route('/books')

.get(function(req,res){

var query = req.query

Book.find(query,function (err, books) {

if (err)

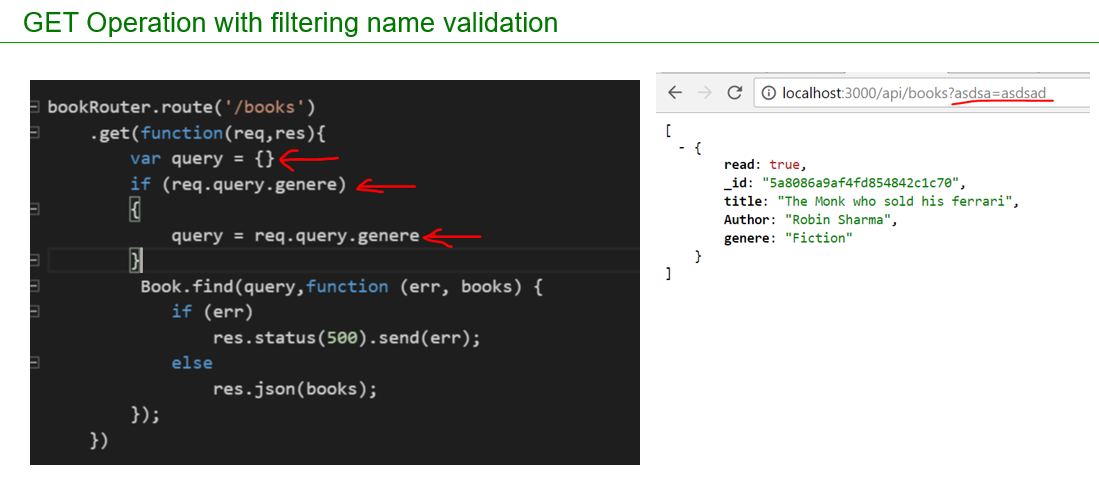
res.status(500).send(err);

else

res.json(books);

});

})

1. GET API with filter name check
   1. 
   2. Place the below code appropriately in app.js ( do not overwrite entire app.js)

bookRouter.route('/books')

.get(function(req,res){

var query = {}

if (req.query.genere)

{

query = req.query.genere

}

Book.find(query,function (err, books) {

if (err)

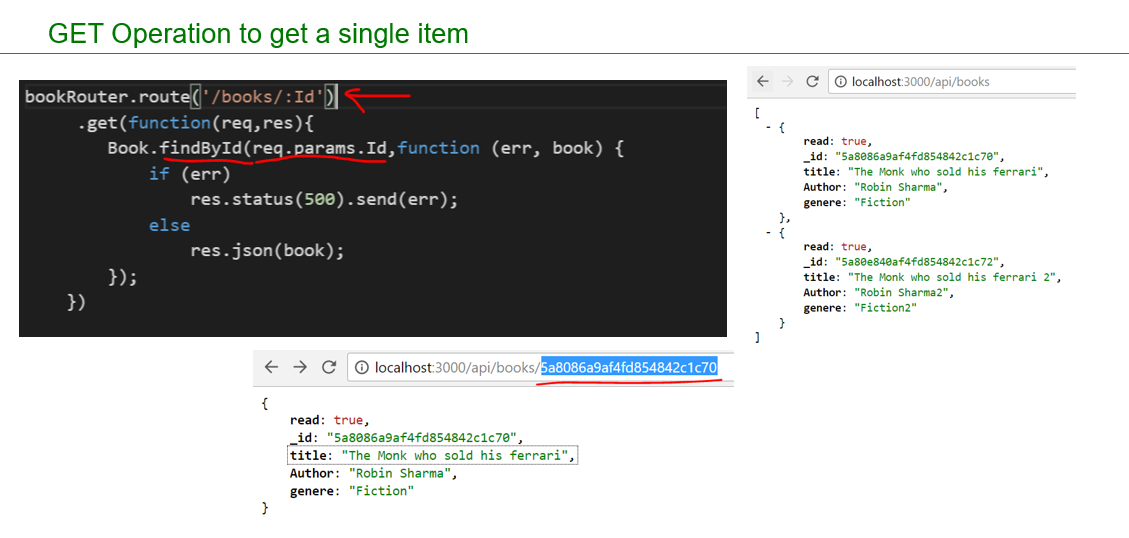
res.status(500).send(err);

else

res.json(books);

});

})

1. GET API to get a single item
   1. 
   2. Place the below code appropriately in app.js ( do not overwrite entire app.js)

bookRouter.route('/books/:Id')

.get(function(req,res){

Book.findById(req.params.Id,function (err, book) {

if (err)

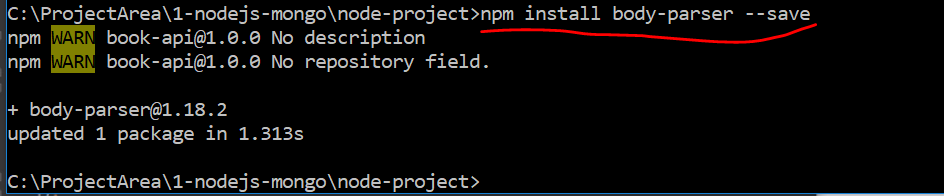
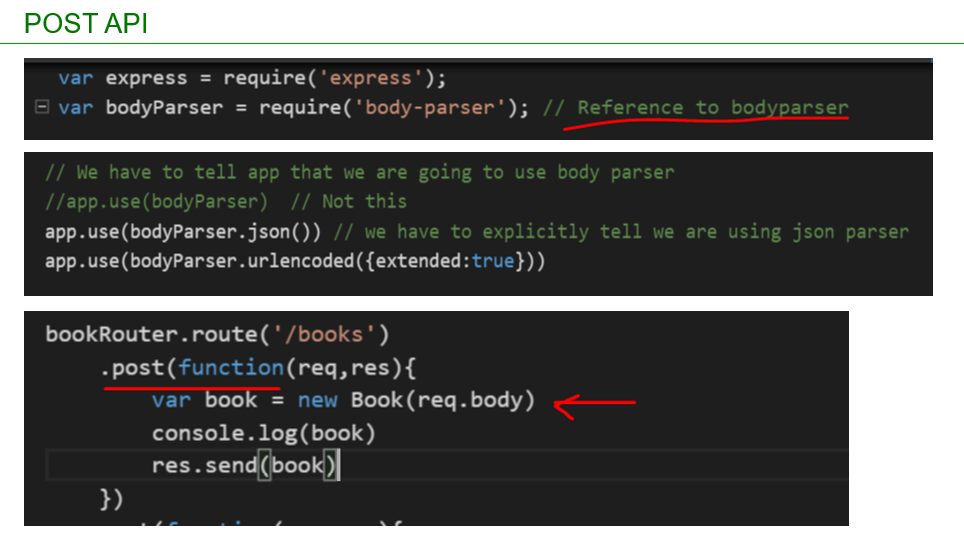
res.status(500).send(err);

else

res.json(book);

});

})

1. Posting data
   1. Install body parser package which is required to read the posted data and convert them to json
   2. 
   3. POST API code changes in app.js
   4. 
   5. Place the below code appropriately in app.js ( do not overwrite entire app.js)

var bodyParser = require('body-parser'); // Reference to bodyparser

// We have to tell app that we are going to use body parser

//app.use(bodyParser) // Not this

app.use(bodyParser.json()) // we have to explicitly tell we are using json parser

app.use(bodyParser.urlencoded({extended:true}))

bookRouter.route('/books')

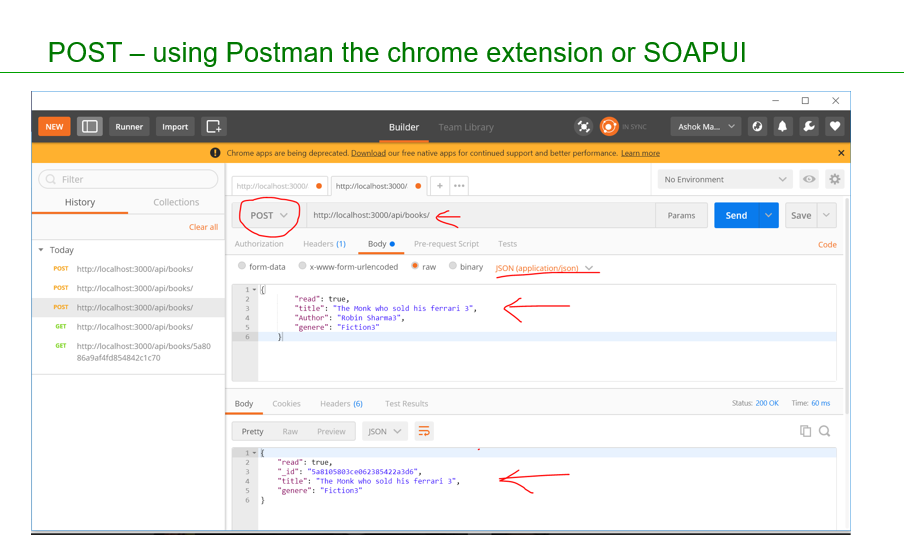
.post(function(req,res){

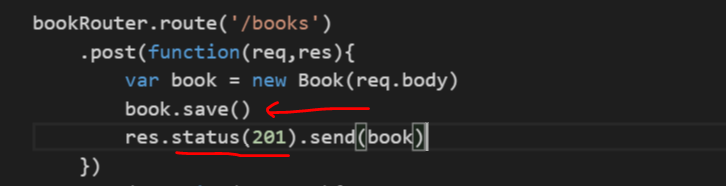
var book = new Book(req.body)

console.log(book)

res.send(book)

})

* 1. Insert below json into DB
     1. {
     2. "title": "The Monk who sold his ferrari",
     3. "author": "Robin Sharma",
     4. "genre": "Fiction",
     5. "read": false
     6. }
  2. Execute the code using postman or soapui
  3. 

1. Posting data – Save to Mongodb
   1. 
   2. Place the below code appropriately in app.js ( do not overwrite entire app.js)

bookRouter.route('/books')

.post(function(req,res){

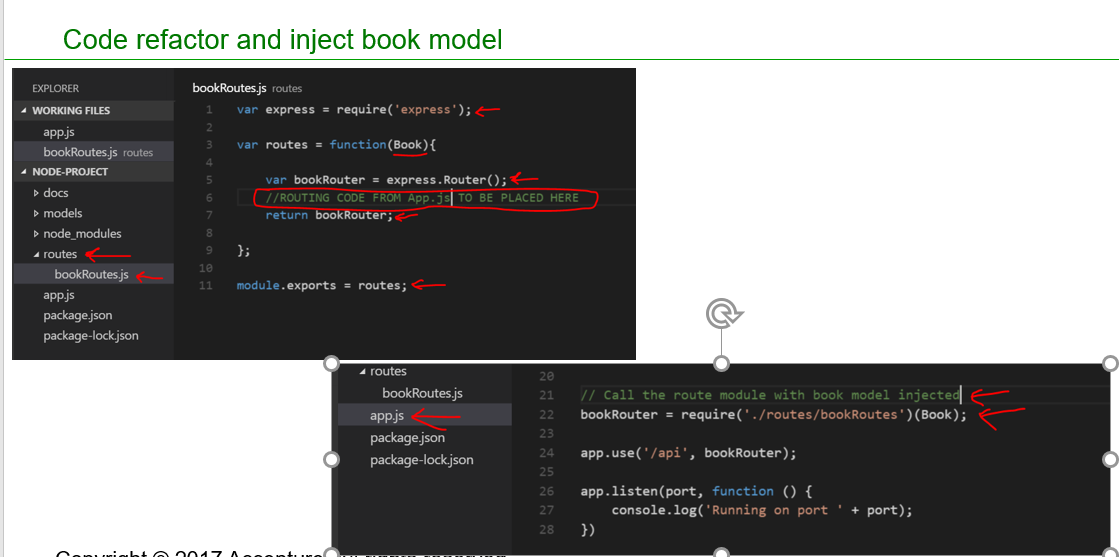
var book = new Book(req.body)

book.save()

res.status(201).send(book)

})

1. Posting data – Injecting model with code cleanup
   1. As our code in app.js is growing we need move some code(routing API code) to a separate file and inject books into it
   2. Create a file as shown below



* 1. Place the below code appropriately in bookRoutes.js

var express = require('express');

var routes = function(){

var bookRouter = express.Router();

bookRouter.route('/books')

.post(function(req,res){

var book = new Book(req.body)

book.save()

res.status(201).send(book)

})

.get(function(req,res){

var query = {}

if (req.query.genere)

{

query = req.query.genere

}

Book.find(query,function (err, books) {

if (err)

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

else

res.json(books);

});

})

bookRouter.route('/books/:Id')

.get(function(req,res){

Book.findById(req.params.Id,function (err, book) {

if (err)

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

else

res.json(book);

});

})

return bookRouter;

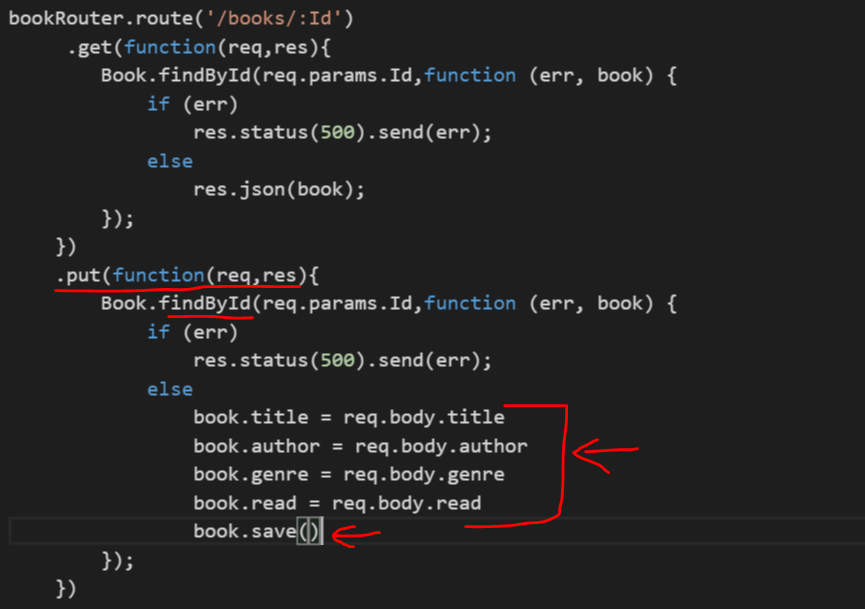
};

module.exports = routes;

1. Place the below code appropriately in app.js ( do not overwrite entire app.js)

// Call the route module with book model injected

bookRouter = require('./routes/bookRoutes')(Book);

1. PUT API
   1. 
   2. Place the below code appropriately in bookRoutes.js

bookRouter.route('/books/:Id')

.get(function(req,res){

Book.findById(req.params.Id,function (err, book) {

if (err)

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

else

res.json(book);

});

})

.put(function(req,res){

Book.findById(req.params.Id,function (err, book) {

if (err)

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

else

book.title = req.body.title

book.author = req.body.author

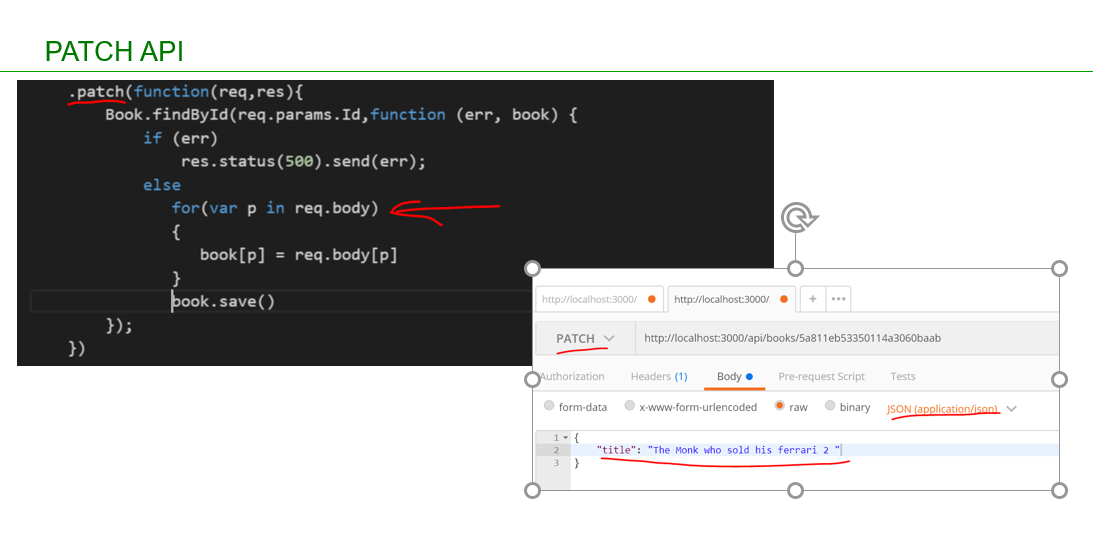
book.genre = req.body.genre

book.read = req.body.read

book.save()

});

})

1. PATCH API
   1. 
   2. Place the below code appropriately in bookRoutes.js

.patch(function(req,res){

Book.findById(req.params.Id,function (err, book) {

if (err)

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

else

for(var p in req.body)

{

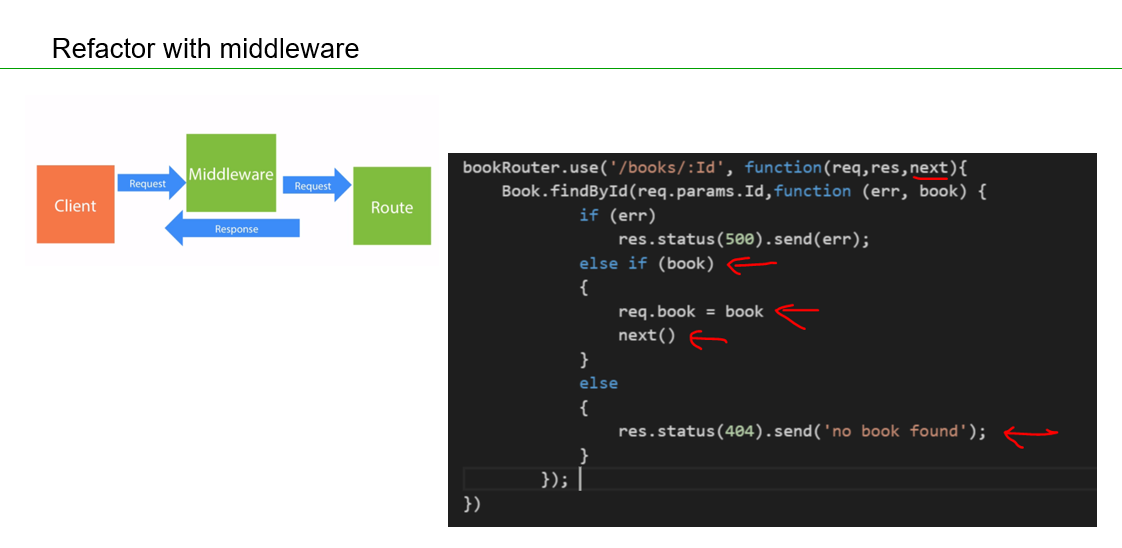
book[p] = req.body[p]

}

book.save()

});

})

1. Refactor with middleware
   1. 
   2. Place the below code appropriately in bookRoutes.js

// Middle Ware

bookRouter.use('/books/:Id', function(req,res,next){

Book.findById(req.params.Id,function (err, book) {

if (err)

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

else if (book)

{

req.book = book

next()

}

else

{

res.status(404).send('no book found');

}

});

})

bookRouter.route('/books/:Id')

.get(function(req,res){

res.json(req.book);

})

.put(function(req,res){

req.book.title = req.body.title

req.book.author = req.body.author

req.book.genre = req.body.genre

req.book.read = req.body.read

req.book.save()

res.json(req.book)

})

.patch(function(req,res){

if(req.body.\_id)

delete req.body.\_id

for(var p in req.body)

{

req.book[p] = req.body[p]

}

req.book.save()

res.json(req.book)

})

1. Remove API

