

COMP 1842

Coursework CRUD system part 1

Matt Prichard

This is for the coursework

These notes form the first part of the CRUD system you are making for your course work.

Just getting all this working is no small achievement so go slow.

Introduction

This week we will create our sever folder/file structure




Install the necessary dependencies

Set up our basic Node server.



Follow very carefully, if it all goes wrong you can just bin it and start again.

My materials will be based on me working on Virtual Desktop

Finished system demo

 Words  New  Test


Show Word

 German	www
 English	bbb

[Edit word](#)

<http://localhost:8080/words/>

Based on sitepoint book

sitepoint

BlogDiscordForumLibraryDashboardAccount

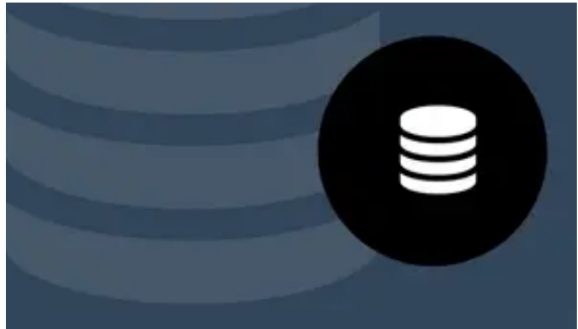
DetailsCreatorContentReviewsFAQ

Build a Basic CRUD App with Vue.js, Node and MongoDB

★★★★☆ 4.2 average rating (5 votes)

Created by **James Hibbard**


Published by **SitePoint**



**Build a Basic CRUD App
with Vue.js, Node and**

Install nodemon globally

```
C:\Users\pm76\fakeg>npm install -g nodemon  
changed 32 packages in 2s  
  
3 packages are looking for funding  
  run `npm fund` for details  
  
C:\Users\pm76\fakeg>_
```



Run this command from your G
Drive

`npm install -g nodemon`

```
C:\Users\pm76\fakeg>nodemon -v  
2.0.21
```

We can check it has installed
`nodemon -v`



nodemon

nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.

nodemon does **not** require *any* additional changes to your code or method of development.

nodemon is a replacement wrapper for `node` . To use `nodemon` , replace the word `node` on the command line when executing your script.

<https://www.npmjs.com/package/nodemon>

Creating the Node Back End

This week and next week we'll create a RESTful API for our Vue front end to consume and test it using PostCode, a VS code plugin for testing APIs

We won't want to go into too much detail on how a RESTful API works this week.

Directory structure pt 1

On your G Drive create folder 'vocab-builder'

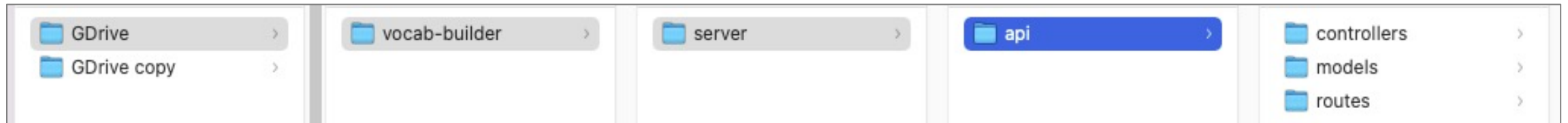
Please follow the naming conventions and structure exactly

Inside this folder create folder 'server'

Directory structure pt2

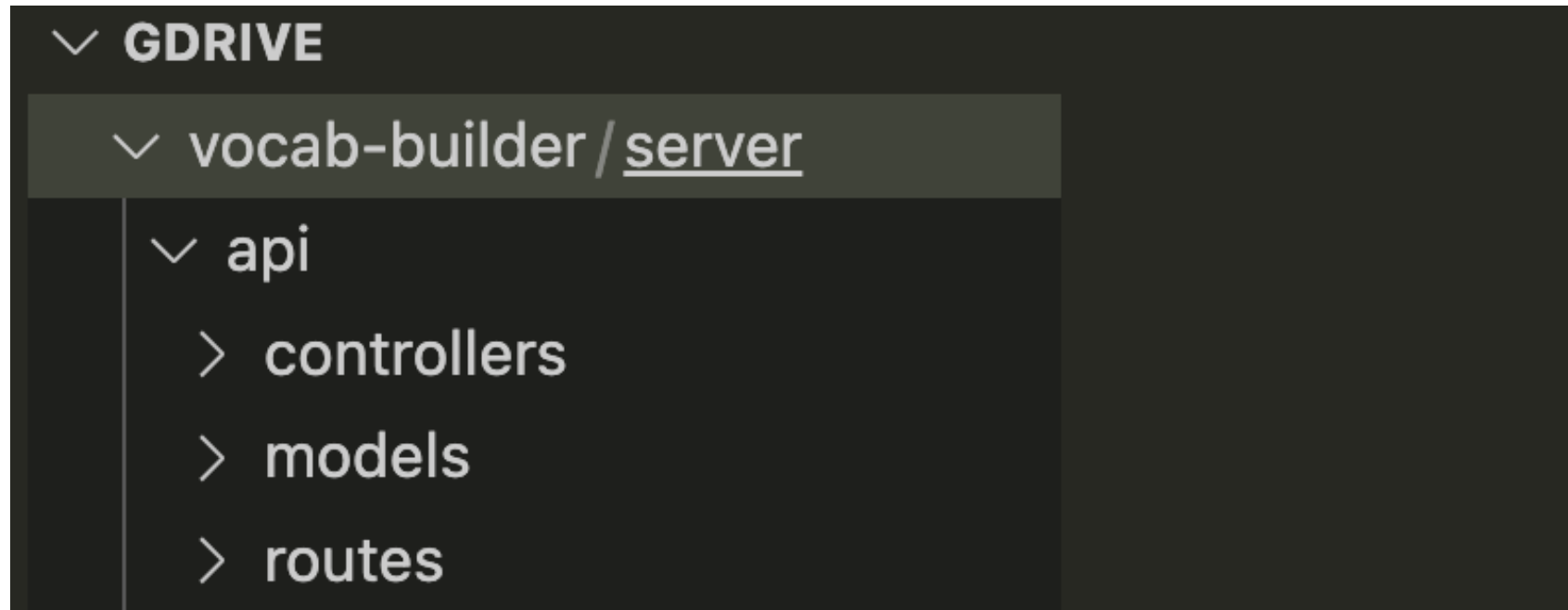
Inside server create 'api'

Inside api create 'controllers', 'models', 'routes'



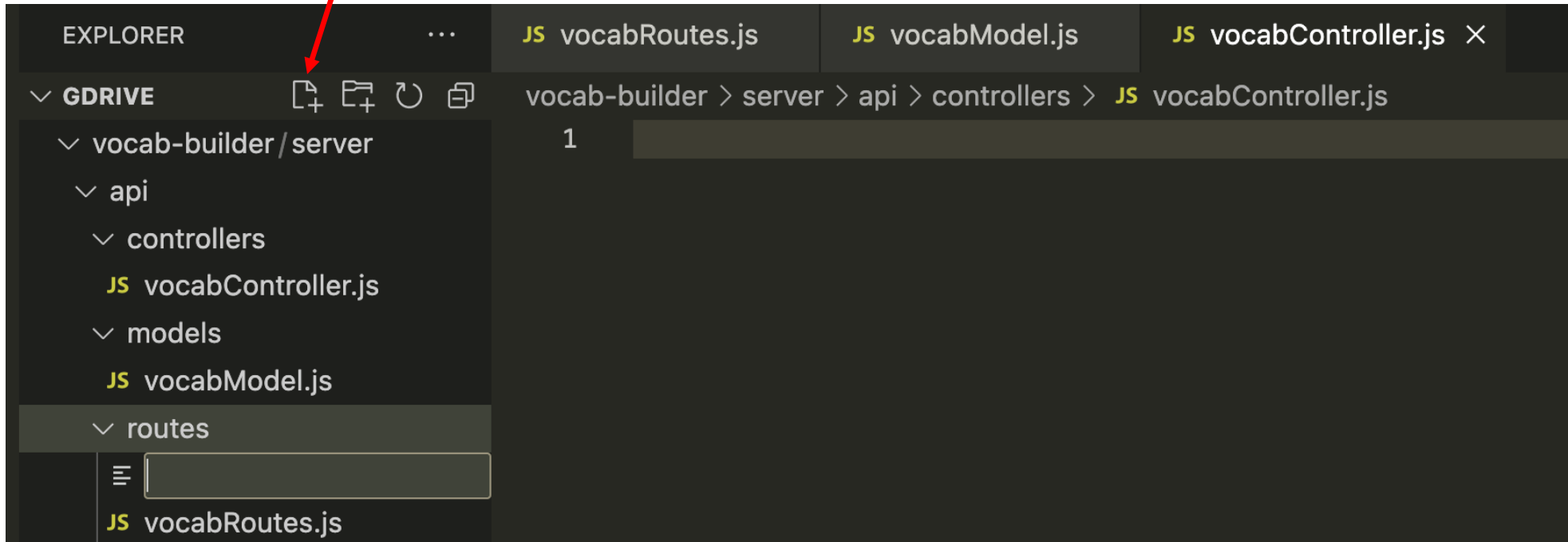
Directory structure pt3

Now open folder 'vocab-builder' from within VS code



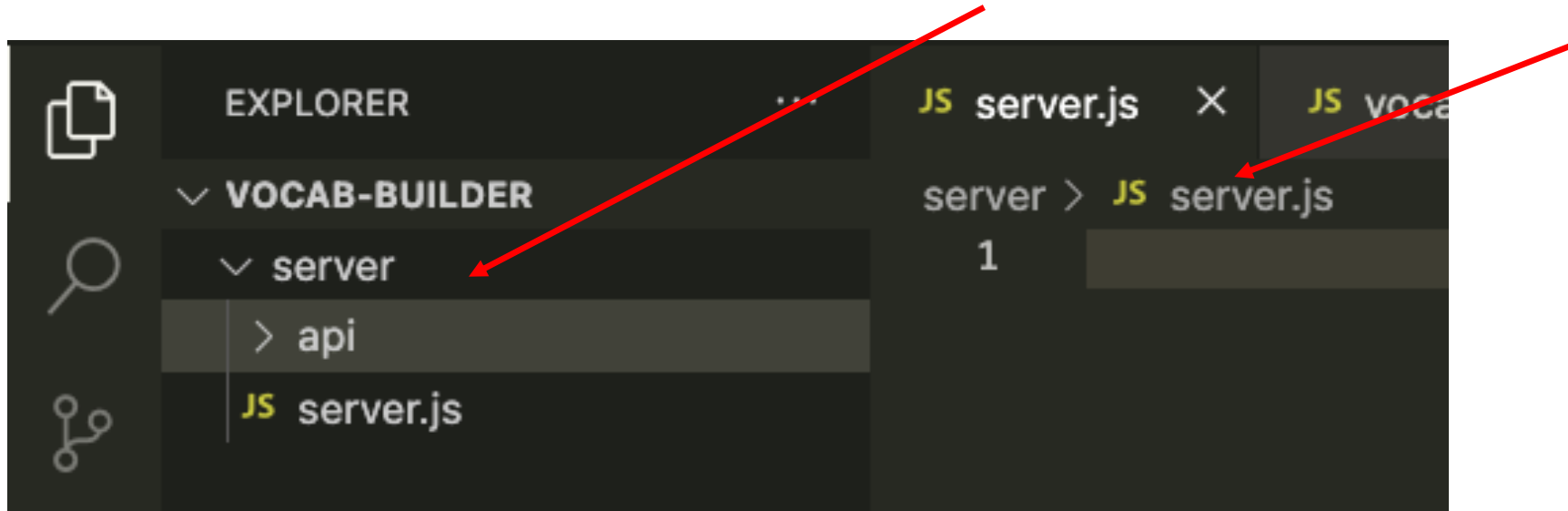
Create files

Using the new file icon create the following 3 js files in their corresponding folders. Be very careful with spelling and Caps



Create files pt2

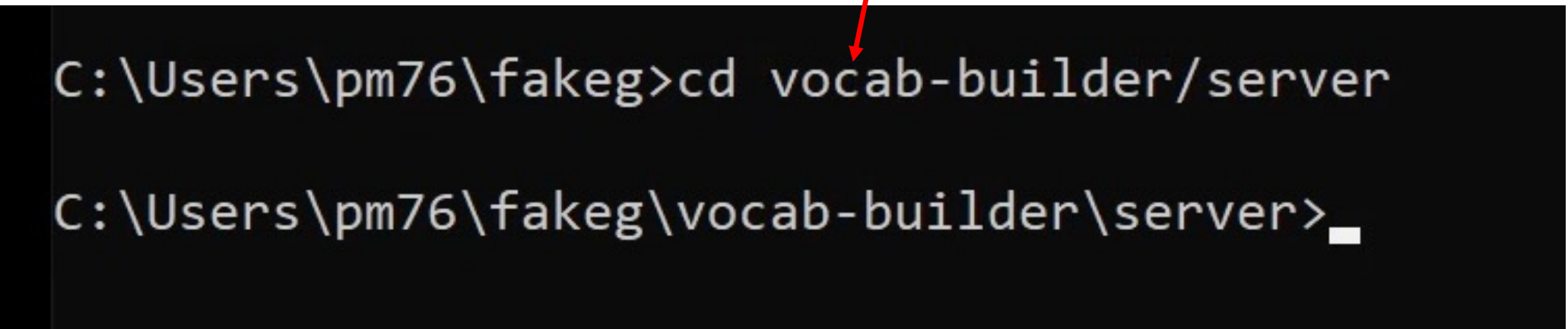
Making sure you are now in the server folder, create server.js file



Leaving VS code open go to the node js command prompt

Node command prompt

Using cd navigate to vocab-builder/server in the command prompt

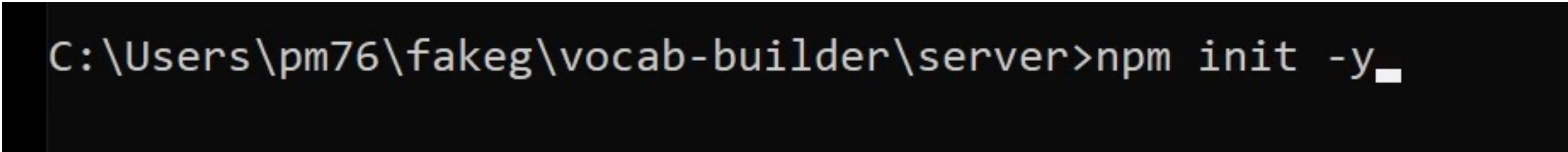


```
C:\Users\pm76\fakeg>cd vocab-builder/server  
C:\Users\pm76\fakeg\vocab-builder\server>_
```

Node command prompt

Once in the server directory run

```
npm init -y
```

A screenshot of a Windows command prompt window with a black background and white text. The text shows the current directory path 'C:\Users\pm76\fakeg\vocab-builder\server' followed by the command 'npm init -y' and a cursor. The prompt character is a greater-than sign '>'.

```
C:\Users\pm76\fakeg\vocab-builder\server>npm init -y_
```

npm init initialises the project and creates our package.json file

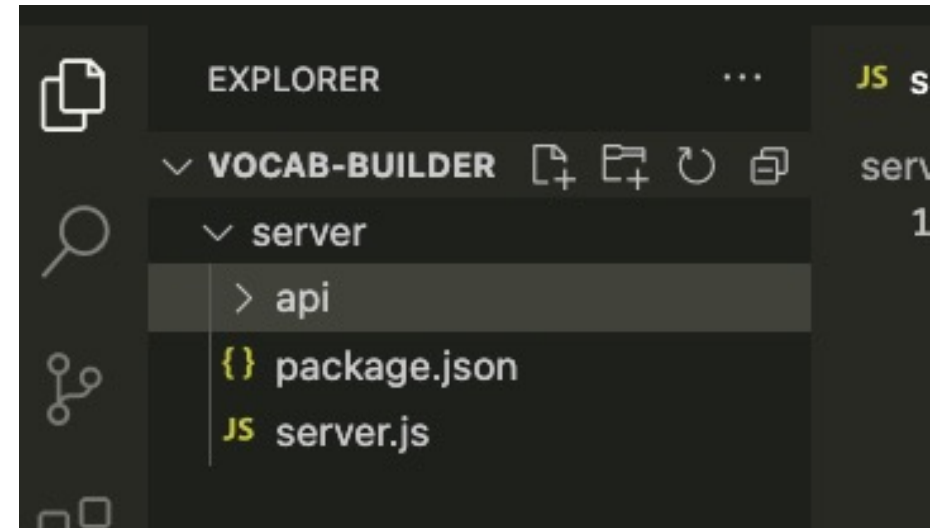
npm init

When you run the command you should see something like this, with the package file appearing in your VS code window

```
Wrote to /Users/matt/GDrive/vocab-builder/server/package.json:
```

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

with something like Standard)



Structure at this point

Your folder and file structure should match this, if not go no further.

```
├── server
│   ├── api
│   │   ├── controllers
│   │   │   └── vocabController.js
│   │   ├── models
│   │   │   └── vocabModel.js
│   │   └── routes
│   │       └── vocabRoutes.js
│   ├── package.json
│   └── server.js
```

Install the dependencies

- For our API, we'll be using the following libraries:
- [express](#) —a small framework for Node that provides many useful features, such as routing and middleware.
- [cors](#) —Express middleware that can be used to enable [CORS](#) in our project.
- [body-parser](#) —Express middleware to parse the body of incoming requests.
- [mongoose](#) —a MongoDB ODM (the NoSQL equivalent of an ORM) for Node. It provides a simple validation and query API to help you interact with your MongoDB database.

Install the dependencies

Making sure you are still in the server directory run this command in the node command prompt – notice we are specifying an exact version of mongoose

```
npm i express cors body-parser mongoose@6.2.4
```

```
[matt@Matts-MacBook-Pro server % npm i express cors body-parser mongoose@6.2.4
```

```
added 89 packages, and audited 90 packages in 7s
```

```
11 packages are looking for funding  
  run `npm fund` for details
```

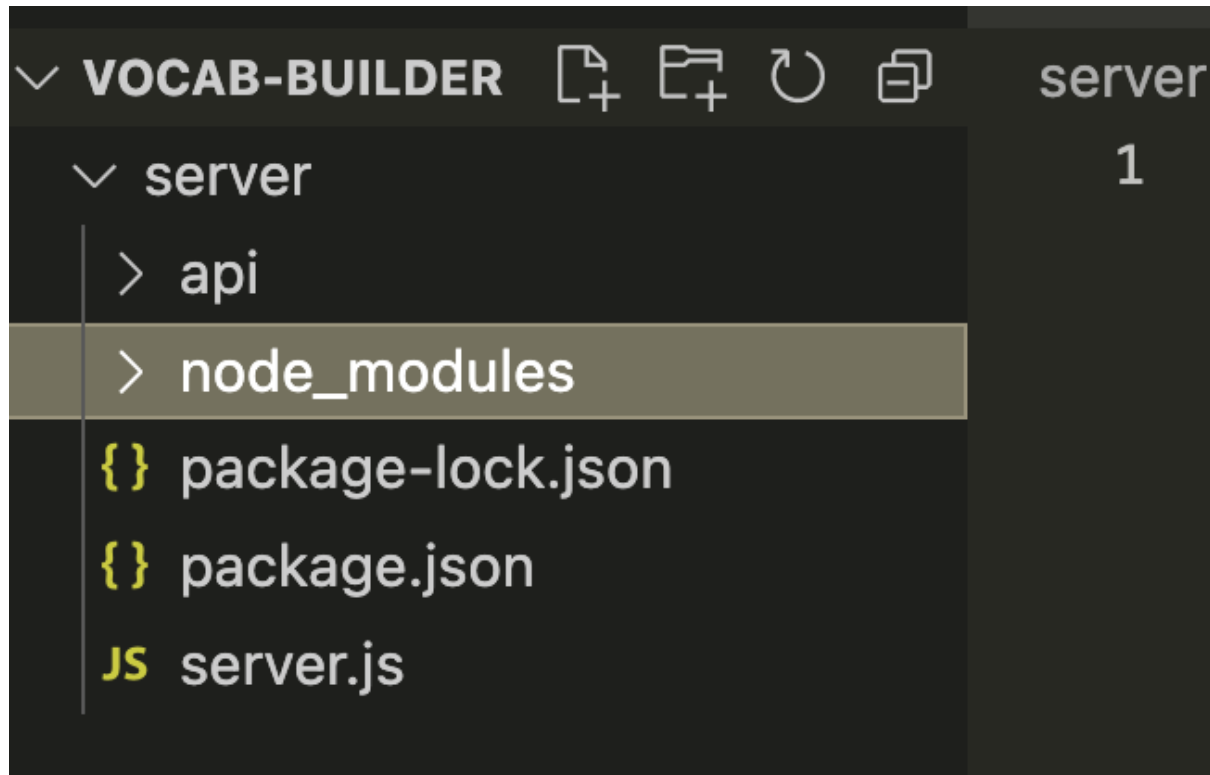
```
1 high severity vulnerability
```

```
To address all issues, run:  
  npm audit fix --force
```

```
Run `npm audit` for details.
```

In VS code

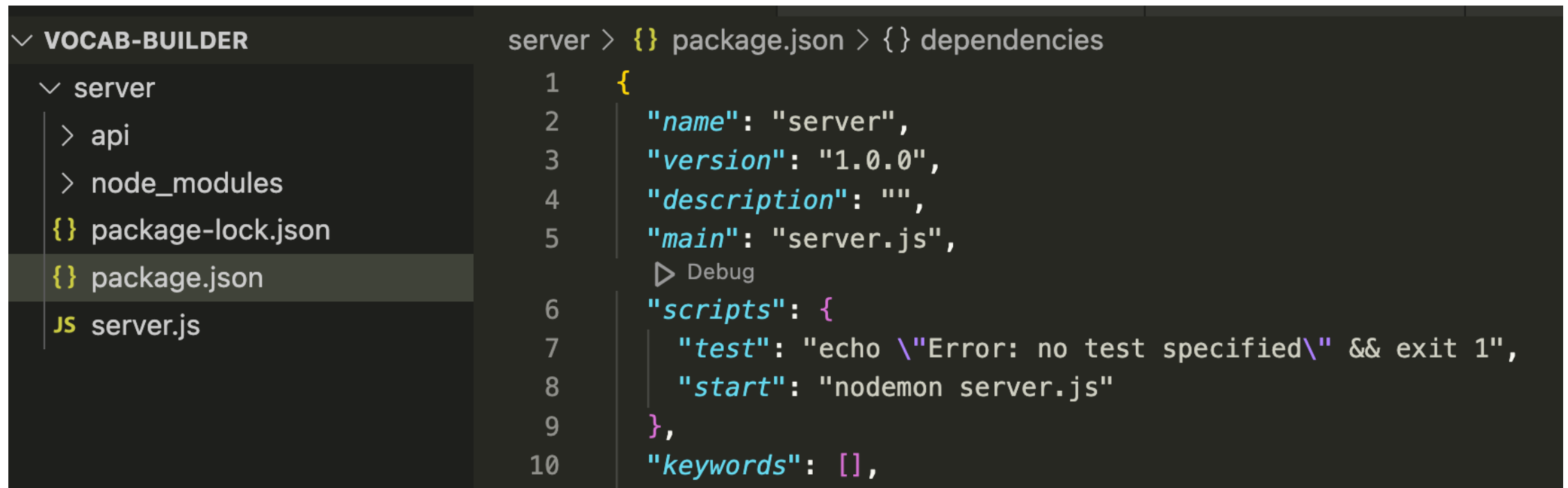
You should now see a new folder called `node_module` that contains many folders including the one we just asked to install



package.json

Open the main package.json file in the server folder and change line8 to read

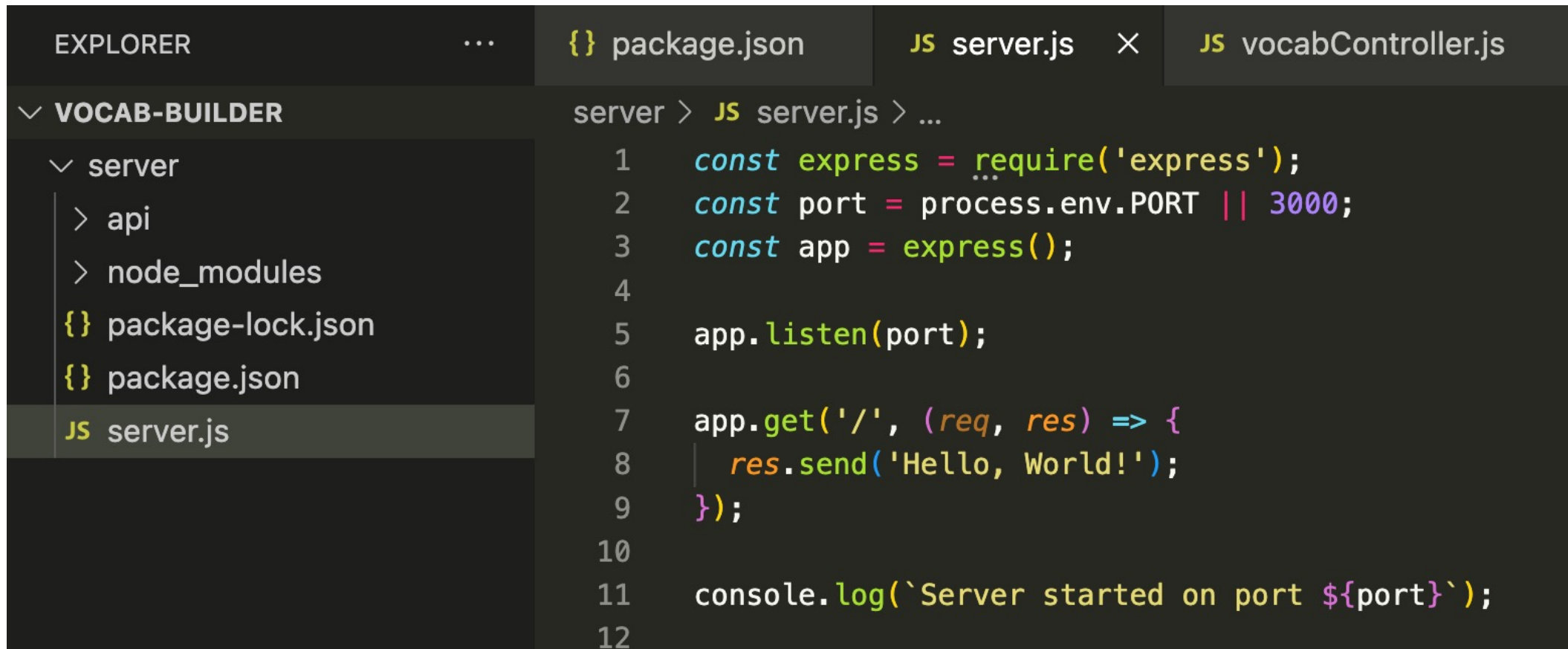
`"start": "nodemon server.js"` now save changes



```
server > {} package.json > {} dependencies
1  {
2    "name": "server",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "start": "nodemon server.js"
9    },
10   "keywords": [],
```

Create the server

Copy this code carefully into the server.js file and save



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project named 'VOCAB-BUILDER' with a 'server' folder containing 'api', 'node_modules', 'package-lock.json', 'package.json', and 'server.js'. The 'server.js' file is selected and open in the editor. The editor shows the following JavaScript code:

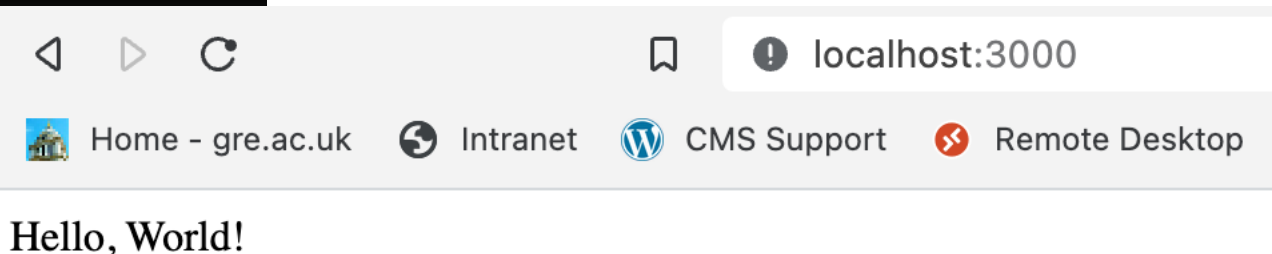
```
server > JS server.js > ...
1  const express = require('express');
2  const port = process.env.PORT || 3000;
3  const app = express();
4
5  app.listen(port);
6
7  app.get('/', (req, res) => {
8    res.send('Hello, World!');
9  });
10
11 console.log(`Server started on port ${port}`);
12
```

run the server

Go into the terminal and run `npm run start`. You should see a message that the server has started on port 3000. If you visit `http://localhost:3000/`, you should see “Hello, World!” displayed.

```
[matt@Matts-MacBook-Pro server % npm run start
> server@1.0.0 start
> nodemon server.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server started on port 3000
```



nodemon working

Leave the server running, go back to VS code and change the message to include your name, when you save the file. nodemon automatically restarts the server for you without having to run the command again. Control C to stop

```
{} package.json  JS server.js  JS vocabController.js

server > JS server.js > ...
1  const express = require('express');
2  const port = process.env.PORT || 3000;
3  const app = express();
4
5  app.listen(port);
6
7  app.get('/', (req, res) => {
8    res.send('Hello, World matt!');
9  });
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
Server started on port 3000
```


Enough for one day.

There is a great deal that can go wrong getting this far, I've had all sorts of difficulties with version control and constantly updated dependencies etc.

If it all goes horribly wrong just bin it all and start again, you won't break anything.



