# Lab 1: Contacts Web App – API

In this lab, you will start building the Contacts web app which we will continue to work on in the course. This web app stores and manages a user’s list of contacts. The objective of this lab is to get started building the API portion of the web app. Follow the steps below.

## Steps

1. Create a folder and name it “contacts-api”.
2. Open the folder. Open Windows “cmd”. To do this, once inside the folder, select the address bar in the window and type cmd. The black terminal window should appear.
3. Enter “npm init” in the cmd window. Follow along the prompts and when asked for the entry point file name, enter “app.js”.
4. Open the contacts-api folder in VS Code.
5. Open a terminal window
6. Enter “npm install express”.
7. Add a “.gitignore” file in the project (ensure there is a “.” At the beginning of the filename). Find a sample online to add to the .gitignore file.
8. Add a “public” folder and add an “images” folder inside the public folder. Add a blank “index.html” file inside the images folder.
9. Add a “routes” folder.
10. Add an “app.js” file.
11. Add the following code to the app.js file.

import express from 'express';

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

const app = express();

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use(express.static('public'));

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

res.send('Hello Express!');

});

app.listen(port, () => {

console.log(`Example app listening on port ${port}`);

});

1. Open the package.json file. Add the following lines in the correct place:  
     
   "type": "module"  
   "start": "node app.js"
2. Test the app to ensure it works ok. If it is not displaying the hello message in the browser, review the steps and troubleshoot.
3. Inside the routes folder, add “contacts.js”. Add the following code to the file:

import express from 'express';

const router = express.Router();

router.get('/', (req, res) => {

res.send('Contacts route');

});

export default router;

1. Open the app.js file. Add the following lines in the correct place:  
     
   import contactsRouter from './routes/contacts.js';  
   app.use('/api/contacts', contactsRouter);
2. In the app.js file, also delete the app.get(…) code snippet.
3. Test the app now and verify the contacts route works.
4. Up to this point in the course, it has been a hassle to always stop and start the app when we make code changes and want to test while the app is running. We are now going to install “nodemon”. When you run an app with nodemon, if you save a change to a project file, nodemon will detect this change and automatically restart the app so the code change takes effect immediately in the running app. To install nodemon globally, enter the following command in the terminal:  
   npm install -g nodemon
5. Open the package.json file. Change “node app.js” to “nodemon app.js”. Try saving a change in the contacts response message while the app is running to see nodemon doing its thing.
6. Add the following code in the contacts.js file to start adding CRUD route handling functionality:  
     
   // Get all contacts

router.get('/all', (req, res) => {

res.send('All contacts');

});  
  
// Get a contact by id

router.get('/:id', (req, res) => {

  const id = req.params.id;

  res.send('Contact by id ' + id);

});

// to-do: add post, put, and delete routers

1. In the contacts.js code, add routers for:
   1. router.post(‘/create’, …)
   2. router.put(‘/update’, …)
   3. router.delete(‘/delete’, …)
2. Open Thunder Client and test each route handler: GET (all and by id), POST, PUT, DELETE. If you don’t get a response for each, review steps and troubleshoot. Otherwise, success!