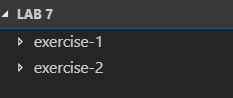
**COMP 3133 – Full Stack Development II – Lab06**

* Express Generator + Router
* Angular First Application

**Developer Note:**When working on your exercises, please create separate folder for your work. This way you won’t putting all your code in the same file, which can pollute the global name space. In short, it will prevent you from overwriting your own work and causing your code to compile incorrectly.

Organize your folder structure in this way.



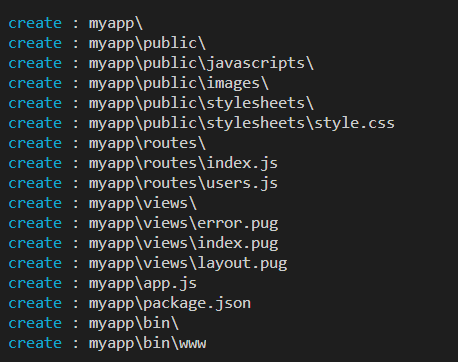
**Exercise #1 – Express Generator**

1. Create a folder named Lab06\_StudentID
2. Open a command prompt create a directory for **exercise-1**
3. Open Visual Studio Code and open the folder **exercise-1**
4. Run the following to install the express application generator

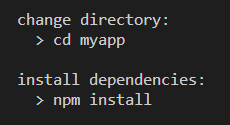


1. The following creates an Express app named **myapp**. The app will be created in a folder named **myapp** in the current working directory and the view engine will be set to [Pug](https://pugjs.org/)



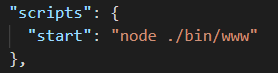
The result will be the following file skeleton application structure being created in ***myapp***.   
 

1. Change directory to the ***myapp*** and then install the application dependencies



1. Navigate to the ***package.json*** file and review the following scripts section. A CLI ***start script command*** has been generated so that node will execute a file from the bin/www folder.

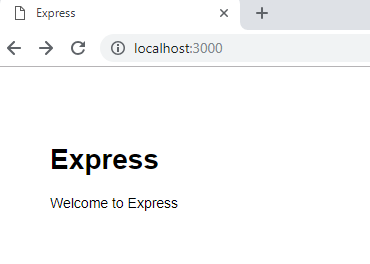
<https://docs.npmjs.com/cli/run-script>



1. Run the npm start CLI command to start the web server on port 3000



1. Navigate the browser to the following URL and view the response from node.



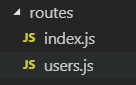
**Exercise #2 – Routes + Body Parser**

* 1. In the same ***exercise-1*** directory, stop the running node application and install the following body-parser middleware.

<https://www.npmjs.com/package/body-parser>



* 1. In the directory structure navigate to the router folder.



* 1. In the ***user.js file*** add the following code to include the ***body-parser*** middleware and set up the router to use the body-parser. Modify the code to reflect the following:

Text

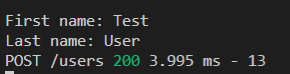
Description automatically generated

* 1. Remember to stop the running application and then rebuild the application with ***node app.js*** and then run the application with ***npm start***
  2. Using POST API Testing Tool build some post parameters and the following POST URL

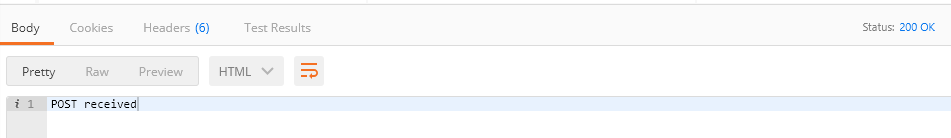
Graphical user interface, application

Description automatically generated

* 1. Write a route in the ***user.js*** that handles the POST submission and uses ***body-parser*** middleware to pull the request parameters from the form and display them in the console log.



The response sent to client i.e. POSTMAN from the user route should be the following ‘POST received!’ message:



**Exercise #3 – Angular CLI + First Application**

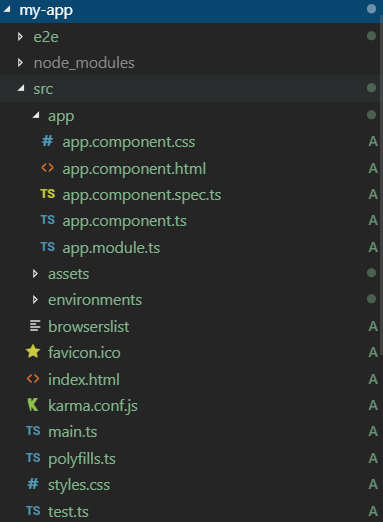
1. Open a command prompt create a directory for **exercise-2**
2. Open Visual Studio Code and open the folder **exercise-2**
3. At the command line using ***npm*** to install the ***Angular CLI*** globally.



1. Create a new project and default angular application named ***myapp*** by running the following CLI command.



The following angular application file structure will be generated.



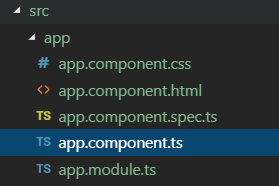
1. Navigated to the ***my-app*** directory and launch the application using the CLI commands



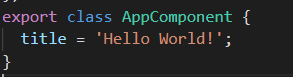
1. Once the Angular development server is loaded it will launch the site on localhost with the following page.



1. Navigate to the root component ***app.component.ts*** and edit your first typescript file.



1. Change the title of the Angular to the following:



The server will automatically track the changes and update the browser page to the following:

