The Unix Command Line

Select the directory you want to upload.

* To create a folder in command line –> mkdir(make directory) Music(folder name).
* To back from the folder –> /cd ..
* To quickly open the terminal in the vs code use –> ctrl `
* If to enter the root directory --> cd ~
* To clear the full terminal use --> clear
* To create a new file in a folder use --> for example we have to create a text file --- (touch Text2.txt).
* To directly open the file in the use 🡪> open image.txt
* To delete files in the folder use 🡪 rm filename.rtf
* To delete all files in the folder at once use 🡪 (rm \*)

# Link for the more command line keyewords <https://www.learnenough.com/command-line-tutorial>

Node.JS

Node provides a JS runtime environment on any computer. It is not a framework. By the help of Node.js we can run our JS file.

It allows us to create our website back-end.

* To check version of node.js – node -v
* To run command written in the JS file using the Node.js we first enter in the directory using the cd.

Node Module-

<https://nodejs.org/docs/latest-v18.x/api/index.html>

* In the 2.2 Native module folder we created a JS folder in which we write a command in Node.js for creating a file and Also the message we write in the message.txt file in the same folder.
* To run the command written in the JS file we first enter into the file directory and write the -- node filename into the terminal.

Example—node index.js

# **npm(Node Package manager)**

* Open the folder directory in the terminal and type –> (npm init -y)command.

It will initialise command for making of the package.json file.

To install Packages of npm - <https://www.npmjs.com/>

* To install npm package in the folder/directory – npm install package\_name
* Then from the documentation write the code EX- 2.3+NPM-folder name.
* Require function is used in the commonJS(CJS)
* import function is used in the ECMAScript(EJS), It provides the same functionality as the commonJS function.
* import xxx from 'superheroes'; --- The string name should be same as the name in the package.json file. Ex- superheroes

# Express.js with Node.js

Express.js is a framework of JS. It allows us to create our website back-end.

Creating an Express server

* Create Directory
* Create index.js file
* Initialise NPM using (npm init -y).
* Install the express package using (npm I express)
* Write Server application in index.js
* Start server.

Then change the “type” in the package.json file to module

Using “type”:”module”.

* We **can not run** our local server just by node index.js file , by this we can only check that our file is executed successfully or not.
* netstat -ano |findstr "LISTENING". By using this command we can find which ports(server) are open in our computer at that time.
* ***We can stop our server / clear our output terminal that is generated after the server request by – (ctrl + c).***

HTTP requests

The purpose of our http is to communicate our client computer to the server side.

* When we make these http request , there are five main words come –

These are:-

* GET – (Getting something from the computer or send request of taking getting resource to the server)
* POST – (Server sending resource )
* PUT –(Replace resource)
* PATCH – (Patch up a resource means for example we can replace the resource just by ourself).
* DELETE –(Delete resource ).

Note: There is a package called (nodemon) by which we can restart our server automatically. By this when we change something in our index.js file then it automatically restart our server.

* For this we can use (nodemon index.js) instead of node index.js.

# **PoSTMAN**

HTTP response status codes includes whether a specific HTTP request has been successfully completed.   
Some grouped responses are:

* Informational responses(100 - 199) –> Hole on
* Successful responses (200-299) 🡪 Here you go
* Redirection messages(300- 399) 🡪 Go away
* Client error responses(400 -499) 🡪 You fucked up
* Server error responses(500-599) 🡪 I fucked up

MDN Docs fro Server INFO—

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

After this we done a exercise in the POSTMAN folder for this we need a POSTMAN app in which we can run our localhost’s and there we can directly get our results. GET, POST, PUT, DELETE,PATCH.

# **Middelwares**

Middle wares are basically acts as the middle man between the server and the user. It uses in ***Authentication , accepting cookies , Pre- Processing, Error, Logging -in***.

1. We have to install the npm package called body-parser. It helps in submitting the details of the form which is created. Like the email id and the password.
2. https://www.npmjs.com/package/body-parser   
   npm I body-parser.

* We have to use POST request b/c it sends the request/ resource from the server to the client.

Example – see it in the Middleware named folder saved in the PC.

Some Middlewares Explanation 🡪

1. import express from 'express';

Imports the Express framework, which simplifies the process of building a web server in Node.js.

1. import morgan from 'morgan';

Imports Morgan, a middleware for logging HTTP requests. This is useful for debugging and monitoring your app.

1. import bodyParser from 'body-parser';

Imports the Body-Parser middleware, which parses incoming request bodies (e.g., form submissions).

1. import { dirname } from "path";

Imports the dirname function from the Path module, which helps in getting the directory name of the current file.

1. import { fileURLToPath } from "url";

Imports fileURLToPath, which is used to convert a file:// URL into a path string (useful in ES Modules).

# **EJS(Embedded JS)**

We can write ejs file using the (**.ejs**) extension.

First we have to install npm package of ejs using 🡪 npm I ejs.

In ejs file we code the same html and then integrate the JS using the **<%= %>.** In JS file instead of app.send 🡪 app.sender(index.ejs,{ }) ;

This sends commands to ejs file and then prints the code written into it.

***🡪 Ejs Tags*** -- <https://github.com/mde/ejs/blob/main/docs/syntax.md>

Code Example refer – 4.1Ejs