1. Develop a web server with following functionalities:

- Serve static resources.

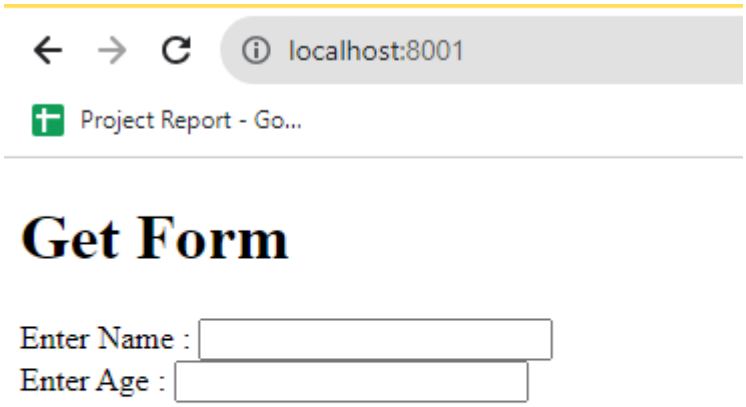
- Handle GET request.

- Handle POST request.

➔ Server static :

```
const http = require('http')
const url = require('url')
const static = require('node-static');

const fileserver = new static.Server('./server');

var server = http.createServer(function(req,res){
    req.addListener('end',function(){
        fileserver.serve(req,res);
    }).resume();
}).listen(8001);
console.log("Listing on port number 8001");
```

output :

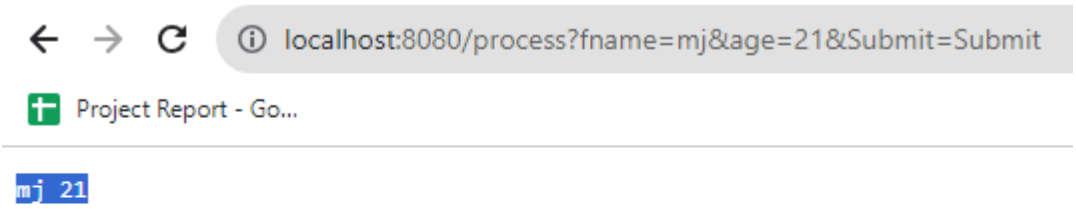


➔ Server-get :

```
var http = require('http')
var fs = require('fs')

var server = http.createServer(function(req,res){
```

```
    console.log("recived url" + req.url);

    if(req.url=="/")
    {
        res.write("hello");
        res.write("hello1");
        res.end();
    }
    else if(req.url=="/list")
    {
        res.write("List");
        res.end();
    }
    else if(req.url=="/index.html" && req.method== 'GET')
    {
        var filename = "./index.html";
        fs.readFile(filename,function(err,data){
            if (err) {
                res.writeHead(404,{'Content-type' : 'text/html'});
                return res.end("404 not found");
            }

            res.writeHead(200,{'Content-type' : 'text/html'});
            res.write(data);
            return res.end();
        });
    }
    else {
        res.write("other pages");
        res.end();
    }

});

server.listen(8080);
console.log("server listing on 8080:");
```

output :

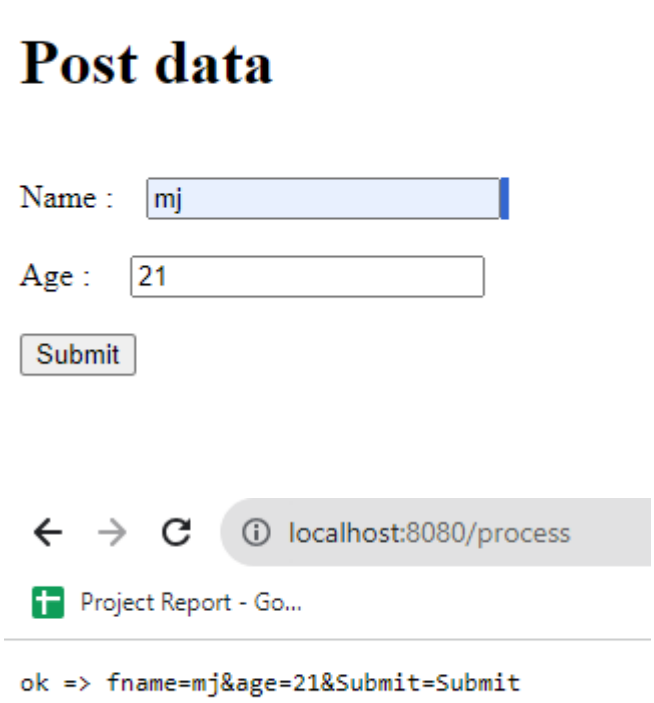


➔ Server-post :

```
var http = require('http')
var fs = require('fs')

var server = http.createServer(function(req,res){
    console.log("recived url" + req.url);

    if(req.url=="/")
    {
        res.write("hello");
        res.write("hello1");
        res.end();
    }
    else if(req.url=="/list")
    {
        res.write("List");
        res.end();
    }
    else if(req.url=="/process" && req.method == 'POST')
    {
        let body = '';
        req.on('data', chunk => {
            body+= chunk.toString();
        });
        req.on('end', () => {
            console.log(body);
            res.end("ok => "+body);
        });
    }
    else {
        res.write("other pages");
        res.end();
    }

});

server.listen(8080);
```

```
console.log("server listing on 8080:");
```

output :



2.Develop nodejs application with following requirements:
- Develop a route "/gethello" with GET method. It displays "Hello NodeJS!!" as response.
- Make an HTML page and display.
- Call "/gethello" route from HTML page using AJAX call. (Any frontend AJAX call API can be used.)

➔ **.js file :**

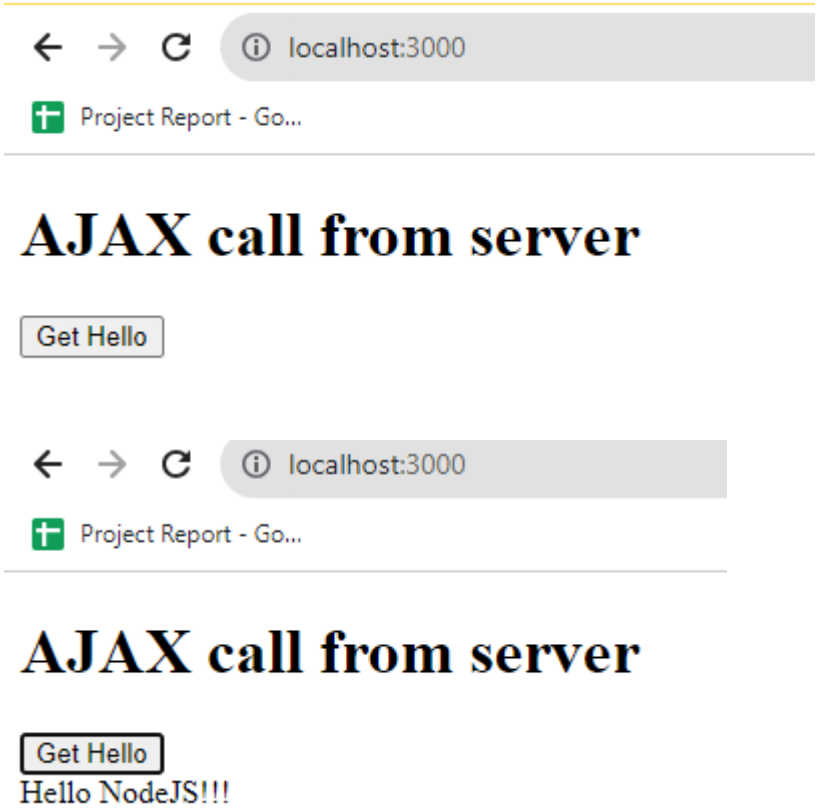
```
const express = require('express');
const app = express();

const path = require('path');

app.get('/gethello', (req, res) => {
    res.send("Hello NodeJS!!!")
});

app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, 'index.html'));
});

app.listen(3000, () => {
    console.log("Server listening on port 3000");
});
```

➔ index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AJAX call</title>
</head>
<body>
    <h1>AJAX call from server</h1>

    <button onclick="getHello()">Get Hello</button>
```

```
    <div id="result"></div>

    <script>
        function getHello() {
            fetch('/gethello')
                .then(responce => responce.text())
                .then(data => {
                    document.getElementById('result').textContent = data;
                })
                .catch(error => {
                    console.log("Error: " + error);
                });
        }
    </script>
</body>
</html>
```

output :



3. Develop a module for domain specific chatbot and use it in a command line application.

➔ Chatbot.js :

```
module.exports.Chatbotreply = function (message) {

    this.bot_age = 25;
    this.bot_name = "Meet-Bot";
    this.bot_university = "vnsgu";
    this.bot_Country = "India";

    message = message.toLowerCase()

    if (message.indexOf("hi") > -1 ||
        message.indexOf("hello") > -1 ||
        message.indexOf("Welcome") > -1) {
        return "hi";
    }
    else if (message.indexOf("age") > -1 &&
        message.indexOf("your")) {
        return "I'm " + this.bot_age;
    }
    else if (message.indexOf("how") > -1 &&
        message.indexOf("are") &&
        message.indexOf("you")) {
        return "I'm fine ^_^";
    }
    else if (message.indexOf("live") > -1 &&
        message.indexOf("where") &&
        message.indexOf("you")) {
        return "I'm live in " + this.bot_Country;
    }

    return "Sorry!, I didn't get it :( ";
}
```

➔ app.js :

```
var Chatbot = require('./Chatbot');
var readline = require('readline');


var r1 = readline.createInterface(process.stdin, process.stdout);
console.log('Welcome user !!!');

r1.setPrompt("You=>");

r1.prompt();

r1.on('line', function (message) {
    console.log('Bot ==> ' + Chatbot.Chatbotreply(message));
}).on('close', function () {
    process.exit(0);
});
```

output :

```
PS D:\NodeJS> node assi_3_app
Welcome user !!!
You=>hi
Bot ==> hi
```

4. Use above chatbot module in web based chatting of websocket.

```
const WebSocket = require('ws')

var http = require('http')

var url = require('url')

var st = require('node-static')

var fileserver = new st.Server('./public')

var httpserver = http.createServer(function(request, response){
    request.on('end',function(){
        var get = url.parse(request.url, true).query;
        fileserver.serve(request,response);
    }).resume();
}).listen(8080,function(){
    console.log((new Date()) +
       'server listening on port 8080');
});

const wss = new WebSocket.Server({server: httpserver});

wss.on('connection', function(ws){
```

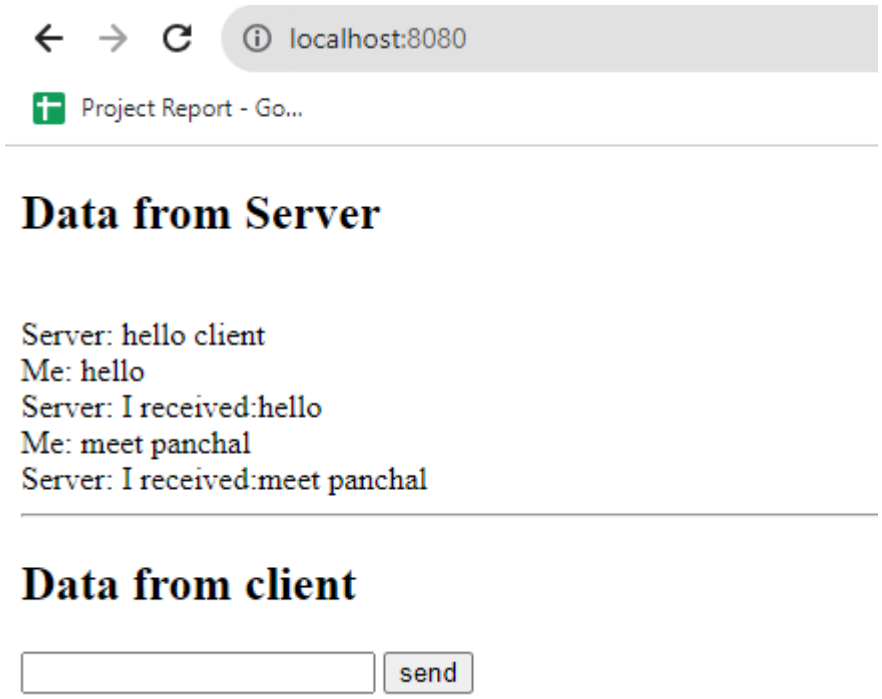
```
    ws.send('hello client')


    ws.on('message', message => {
    console.log('Received message => ${message}')
    ws.send('I received:' + message)
    })


})
```

➔ index.html page:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Web Socket</title>
    <script>
        var ws = new WebSocket('ws://localhost:8080');
        ws.addEventListener('message', function(e){
            var msg = e.data;
            document.getElementById('chatlog').innerHTML+='<br> Server: '+msg;
        });


        function sendMessage(){
            var message = document.getElementById('message').value;
            document.getElementById('chatlog').innerHTML+='<br> Me: '+
message;
            ws.send(message);
            document.getElementById('message').value = '';
```

```
        }
    </script>
</head>
<body>
    <h2>Data from Server</h2>
     <div id="chatlog"></div>


     <hr />


    <h2>Data from client</h2>
     <input type="text" id="message" />
     <input type="button" id="b1" onclick="sendMessage()" value="send" />


</body>
</html>
```

output :

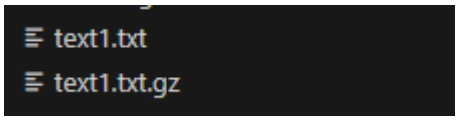


5. Write a program to create a compressed zip file for a folder.

```
var fs = require('fs')
var zlib = require('zlib')

fs.createReadStream('./text1.txt')
    .pipe(zlib.createGzip())
    .pipe(fs.createWriteStream('./text.txt.gz'));

console.log('File compressed..!!');
```

output :

```
PS D:\NodeJS> node asi_5
File compressed..!!
PS D:\NodeJS>
```

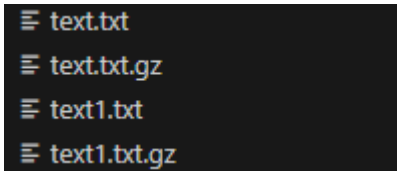


6. Write a program to extract a zip file.

```
//6. Write a program to extract a zip file.

var fs = require('fs')
var unzip = require('zlib')

fs.createReadStream('./text.txt.gz')
    .pipe(unzip.createGunzip())
    .pipe(fs.createWriteStream('./txet.txt'));

console.log('File Decompressed..!!');
```

output :

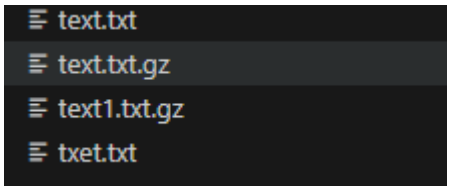
```
PS D:\NodeJS> node assi_6
File Decompressed..!!
PS D:\NodeJS>
```



7. Write a program to promisify fs.unlink function and call it.

```
//7. Write a program to promisify fs.unlink function and call it.

var fs = require('fs/promises')

function readFile(fpath)
{
    return new Promise(function(success,fail)
    {
        fs.unlink(fpath,(err,data) =>
        {
            if(err)
                fail(err)
            else
                success(data)
        })
    })
}

readFile('./text1.txt').then((data)=>{
    console.log(data)
}).catch((err)=>{
    console.log(err)
})
```

output :



8. Fetch data of google page using note-fetch using async-await model.

```
//8. Fetch data of google page using note-fetch using async-await model.

//var fetch = require('node-fetch')

const fetch = (...args) => import('node-fetch').then(({default: fetch}) =>
fetch(...args));


async function asyncajaxawait()
{
  const res = await fetch('https://www.google.com/')
  console.log(res);
}

asyncajaxawait();
```

output :

```
PS D:\NodeJS> node assi_8
Response {
  size: 0,
  [Symbol(Body internals)]: {
    body: Gunzip {
      _writeState: [Uint32Array],
      _readableState: [ReadableState],
      _events: [Object: null prototype],
      _eventsCount: 6,
      _maxListeners: undefined,
      _writableState: [WritableState],
      allowHalfOpen: true,
```

```
      allowHalfOpen: true,
      bytesWritten: 0,
      _handle: [Zlib],
      _outBuffer: <Buffer 00 00 00 00 00 00 00 00 00 0
 00 00 00 00 00 00 00 00 00 00 ... 16334 more bytes>,
      _outOffset: 0,
      _chunkSize: 16384,
      _defaultFlushFlag: 2,
      _finishFlushFlag: 2,
      _defaultFullFlushFlag: 3,
      _info: undefined,
      _maxOutputLength: 4294967296,
```

9.Write a program that connect Mysql database, Insert a record in employee table and display all records in employee table using promise based approach.

```
const mysql = require('nodejs-mysql').default;


const config = {
    host : "localhost",
    user : "root",
    password : "root",
    database : "employee_db"
}


const db = mysql.getInstance(config);


db.connect()
  .then(function(){
    console.log("Connected!!");


    var sql = "INSERT INTO employee (username, password, firstname, lastname, email) VALUES ('meet', 'mj', 'Panchal', 'Meet', 'meet46884@gmail.com')";


    return db.exec(sql);


}).then(function(res){
    console.log(res);
    return db.exec("SELECT * FROM employee");
}).then(function(result){
    for( var i in result){
```

```
        console.log("Username: ", result[i].username + " " + "Password: " +
result[i].password);

        process.exit(0);

    }

}).catch(function(err){

    console.log("ERROR: ", err);

    process.exit(0);

});
```

output :

```
PS D:\NodeJS> node assi_9
Connected!!
OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 2,
  serverStatus: 2,
  warningCount: 0,
```

```
 OkPacket {
   fieldCount: 0,
   affectedRows: 1,
   insertId: 2,
   serverStatus: 2,
   warningCount: 0,
   message: '',
   protocol41: true,
   changedRows: 0
 }
 Username:  meet Password: mj
 PS D:\NodeJS>
```

10. Set a server script, a test script and 3 user defined scripts in package.json file in your nodejs application.

```
{
  "name": "nodejs",
  "version": "1.0.0",
  "description": "",
  "main": "asi_5.js",
  "dependencies": {
    "install": "^0.13.0",
    "node-fetch": "^3.3.1",
    "node-static": "^0.7.11",
    "ws": "^8.13.0"
  },
  "scripts": {
    "server" : "node file.js",
    "test": "node test.js",
    "script1" : "node assi_6.js",
    "script2" : "node assi_7.js",
    "script3" : "node assi_8.js"
  },
  "author": "",
  "license": "ISC"
}
```

output :

```
PS D:\NodeJS> npm run script3

> nodejs@1.0.0 script3
> node assi_8.js

Response {
  size: 0,
```

11. Develop an application to show live cricket score.

```
const express = require('express');
const axios = require('axios');

const app = express();
const port = 3000;

const apiKey = 'd4594015-7cc4-4cd1-9817-610c4768246e'; // Replace with your actual cricapi API key

app.get('/live-score', (req, res) => {
  const cricapiUrl = `https://api.cricapi.com/v1/currentMatches?apikey=d4594015-7cc4-4cd1-9817-610c4768246e&offset=0`;

  axios.get(cricapiUrl)
    .then(response => {
      const liveMatches = response.data.matches.filter(match => match.matchStarted);
      const liveScores = liveMatches.map(match => {
        return {
          id: match.id,
          date: match.date,
          score: match.score,
        };
      });

      res.json(liveScores);
```

```
    })
    .catch(error => {
      console.error('Error fetching live scores:', error.message);
      res.status(500).send('Error fetching live scores.');
    });
});


app.listen(port, () => {
  console.log(`Live cricket score app is running on
http://localhost:${port}`);
});
```