1. Develop a web server with following functionalities:

- Serve static resources.

- Handle GET request.

- Handle POST request.

INDEX.HTML

```html
<!doctype html>
<html>

<body>
    <h1> Get Form</h1>
    <form action="/process" method="POST">
        Enter Name: <input type="text" name="fname" /><br />
        Enter Age: <input type="number" name="age" /><br />
        <button>Save</button>
    </form>
    <br><br>
    <h1> Post Form</h1>
    <form action="/process" method="POST">
        Enter Name: <input type="text" name="fname" /><br />
        Enter Age: <input type="number" name="age" /><br />
        <button>Save</button>
    </form>
</body>

</html>
```

SERVER-GET-POST.JS

```javascript
const http = require('http');
const fs = require('fs');
const port = process.env.PORT || 8000;
const server = http.createServer((req, res) => {
    if (req.url == "/") {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write("hello from the server");
        return res.end();
    }
    else if (req.url == "/process" && req.method == 'POST') {
        let body = '';
        req.on('data', (chunk) => {
            body += chunk;
        });

        req.on('end', () => {
            res.writeHead(200, { 'Content-Type': 'text/plain' });
            res.end('Hello, this is a POST request with body: ' + body);
            console.log(body);


        });
```

```
    }
    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");
            }
            res.writeHead(200, { 'Content-Type': 'text/html' });
            console.log(data);
            res.write(data);
            return res.end();
        });
    }
});
server.listen(port, () => {
    console.log(`server is listening on ${port}`);
});
```

OUTPUT:-

localhost:8000/index.html

Get Form

Enter Name: rutvi
Enter Age: 21
Save

Post Form

Enter Name: rutvi
Enter Age: 22
Save

localhost:8000/process

Hello, this is a POST request with body: fname=rutvi&age=22

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.)

INDEX.HTML

```html
<!DOCTYPE html>
<html>
<head>
    <title>Q2</title>
</head>
<body>
    <button onclick="getHello()">Get Hello</button>
    <p id="response"></p>
    <script>
        function getHello() {
            var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
                if (this.readyState === 4 && this.status === 200) {
                    document.getElementById("response").innerHTML = this.responseText;
                }
            };
            xhttp.open("GET", "/gethello", true);
            xhttp.send();
        }
    </script>
</body>
</html>
```

INDEX.JS

```javascript
const express = require('express');
const app = express();
const port = 3000;

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

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

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

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

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);
});
```

CHATBOT.JS

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


    this.bot_age = 25;
    this.bot_name = "rutvi-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 :( ";
}
```

OUTPUT:

```
PS C:\Users\Dell\Desktop\ICT-3\OSWD(304)\Assignment-1\Q3> npm run start

> q3@1.0.0 start
> nodemon app.js

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
Welcome user !!!
You=>hi
hi
Bot ==> hi
```

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

WEBSOKET.JS

```
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

```
<!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.

FILE.JS

```
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 C:\Users\Dell\Desktop\ICT-3\OSWD(304)\Assignment-1\Q5> npm run start

> q5@1.0.0 start
> file.js

PS C:\Users\Dell\Desktop\ICT-3\OSWD(304)\Assignment-1\Q5>
```

text.txt.gz

6. Write a program to extract a zip file.

FILE.JS

```
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..!!');
```

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

FILE.JS

```
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)
})
```

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

```
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();
```

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 ('rutvi', 'rutvi', 'patel', 'rutvi', 'rutvipatel567@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);
});
```

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": "Q5.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 Q6.js",
    "script2" : "node Q7.js",
    "script3" : "node Q8.js"
  },
  "author": "",
  "license": "ISC"
}
```

11. Develop an application to show live cricket score.

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


const app = express();
const port = 8000;


const apiKey = 'd4594015-7cc4-4cd1-9817-610c4768246e';
```

```
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}`);
});
```