

INDEX

Sr.No	<u>Full Stack Assignment</u>	Page no.	Signature
Q1.	Design front-end application using basic React.		
Q2.	Design front-end applications using functional components of React.		
Q3.	Design back-end applications using Node.js.		
Q4.	Construct web-based Node.js applications.		
Q5.	Implement the concepts of Buffers,Streams and Events.		
Q6.	Implement Multi-Processing in Node.js.		
Q7.	Integrate Angular application with other JavaScript libraries such as Node.js.		
Q8.	Use MongoDB to store data in the Database.		

Full Stack Assignment

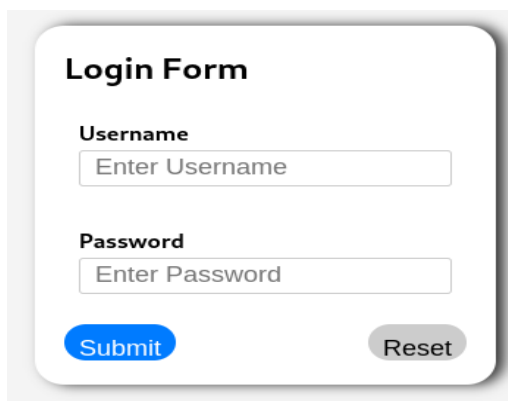
Q1.Design front-end application using basic React.

Program:

```
import './App.css';

function App() {
  return (
    <div>
      <div className="container">
        <form>
          <h1>Login Form</h1>
          <div className="userInput">
            <label>Username</label>
            <input type='text' placeholder='Enter Username' required></input>
          </div>
          <div className="passwordInput">
            <label>Password</label>
            <input type='password'placeholder='Enter Password' required></input>
          </div>
          <div className="submitBtn">
            <input type='submit'></input>
            <input type='reset' id='reset'></input>
          </div>
        </form>
      </div>
    </div>
  )
}
export default App;
```

Output:



Q2.Design front-end applications using functional components of React.

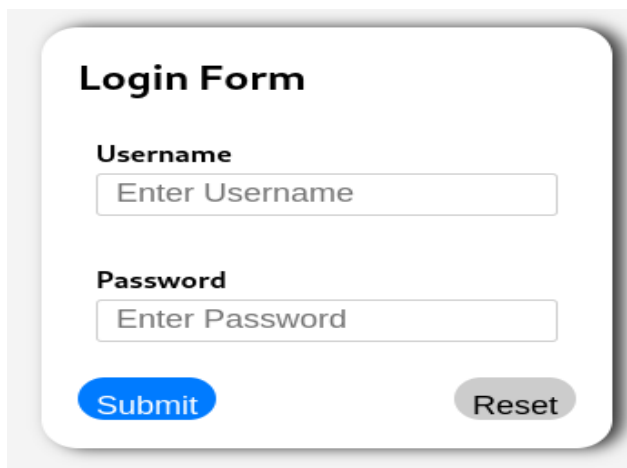
Program:

```
import './App.css';
import Login from './component/Login';
```

Full Stack Assignment

```
function App() {  
  return (  
    <>  
      <Login/>  
    </>  
  )  
}  
export default App;  
  
import React from 'react';  
  
export default function Login() {  
  return (  
  
    <div><div className="container">  
      <form >  
        <h1>Login Form</h1>  
        <div className="userInput">  
          <label>Username</label>  
          <input type='text' placeholder='Enter Username' required></input>  
        </div>  
        <div className="passwordInput">  
          <label>Password</label>  
          <input type='password'placeholder='Enter Password' required></input>  
        </div>  
        <div className="submitBtn">  
          <input type='submit'></input>  
          <input type='reset' id='reset'></input>  
        </div>  
      </form>  
    </div>  
  </div>  
  )  
}
```

Output:



Login Form

Username

Password

Full Stack Assignment

Q3.Design back-end applications using Node.js.

Program:

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

app.get('/', (req, res) => {
  res.send('<h1>Hello, World!</h1>');
});

// Start the server on port 3000
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT} and Server is ready !`);
});
```

Output:

```
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 App.js`
Server is running on port 3000 and Server is ready !
```

Q4.Construct web-based Node.js applications.

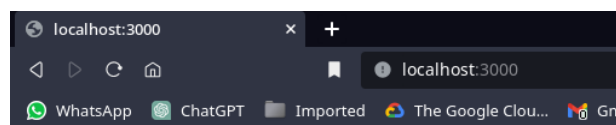
Program:

```
const express = require('express');
//creating an express application
const app = express();

app.get('/',(req,res) => {
  res.send(' <h1> Hello World! </h1>');
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is ready to run on port ${PORT}`);
});
```

Output:



Hello World!

Full Stack Assignment

Q5.Implement the concepts of Buffers,Streams and Events.

Program:

```
const fs = require('fs');
const EventEmitter = require('events');

// Create a custom event emitter instance
const eventEmitter = new EventEmitter();

// Create a readable stream to read from a text file
const readableStream = fs.createReadStream('input.txt');

// Create a Buffer to accumulate data
let buffer = Buffer.alloc(0);

// Listen for the 'data' event from the readable stream
readableStream.on('data', (chunk) => {
  // Concatenate the data chunks into the buffer
  buffer = Buffer.concat([buffer, chunk]);
});

// Listen for the 'end' event from the readable stream
readableStream.on('end', () => {
  // Emit a custom 'processData' event with the processed content
  eventEmitter.emit('processData', buffer.toString());
});

// Define an event listener for the custom 'processData' event
eventEmitter.on('processData', (data) => {
  console.log('Processed Data:');
  console.log(data);
});
```

Output:

node App.js

Processed Data:

- 1.This is Bindaas Banda From MCA.
- 2.This is assignment of full stack.

Q6.Implement Multi-Processing in Node.js.

Program:

```
const cluster = require('cluster');
const numCPUs = require('os').cpus().length;

if (cluster.isMaster) {
  // This code runs only in the master process
  console.log(`Master process (PID: ${process.pid}) is running`);
  // Fork workers for each CPU core
  for (let i = 0; i < numCPUs; i++) {cluster.fork();}
```

Full Stack Assignment

```
cluster.on('exit', (worker, code, signal) => {
  console.log(`Worker process ${worker.process.pid} has exited`);
});
} else {
  console.log(`Worker process ${process.pid} is running`);
}
```

Output:

node multiprocessing.js

Master process (PID: 7027) is running

Worker process 7037 is running

Worker process 7038 is running

Q7. Integrate Angular application with other JavaScript libraries such as Node.js.

Answer:-

Building a Simple Web App with Express(node.js) & Angular

Set Up the Project:

Step 1. Create new directory (mkdir my-angular-app)

Step 2. Navigate to directory (cd my-angular-app)

Step 3. Initialize npm package (npm init)

Step 4. Install Angular & Express (npm install @angular/cli/ express)

Step 5. Create the Angular Client FrontEnd (ng new client)

Step 6. Create the Express server (server.js):-

Code:-

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

```
// handling CORS
```

```
app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin",
    "http://localhost:4200");
  res.header("Access-Control-Allow-Headers",
    "Origin, X-Requested-With, Content-Type, Accept");
  next();
});
```

```
// route for handling requests from the Angular client
```

```
app.get('/api/message', (req, res) => {
  res.json({ message:
    'Hello World ! from the Express server!' });
});
```

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

Full Stack Assignment

Step 7. Update the Angular client to request the express server create new file (“src/app/api.service.ts”):-

Code:-

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class ApiService {
  constructor(private http: HttpClient) { }
  getMessage() {
    return this.http.get(
      'http://localhost:3000/api/message');
  }
}
```

Step 8. Update the app.module.ts to import the HttpClient module:

Code:-

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 9. Use the service to display the angular component (“src/app/app.component.ts”):-

Code:-

```
import { Component, OnInit } from '@angular/core';
import { ApiService } from './api.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```


Full Stack Assignment

```
export class AppComponent implements OnInit {
  title = 'frontEnd';
  message: any;
  constructor(private apiService: ApiService) { };
  ngOnInit() {
    this.apiService.getMessage().subscribe(data => {
      this.message = data;
    });
  }
}
```

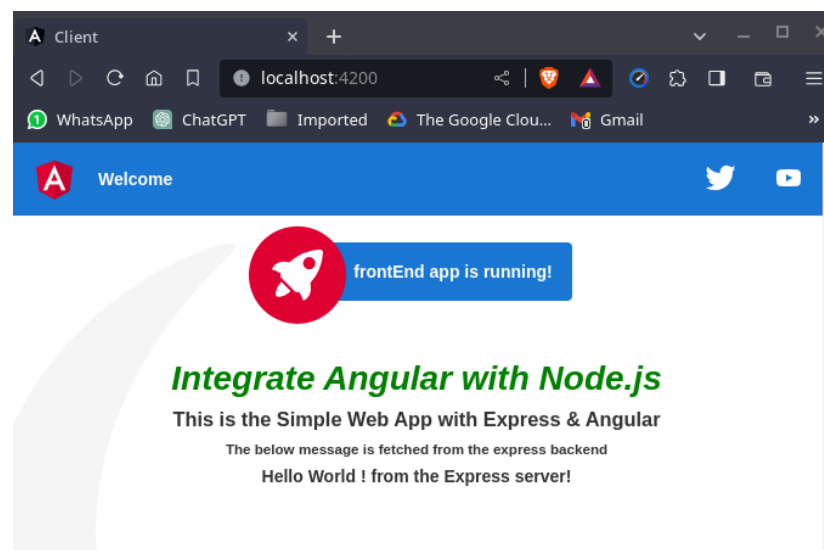
Step 10. Display message on Angular Template (“src/app/app.component.html”):
Code:-

```
<div style="text-align: center;">
  <h1 style="color: green; font-style: italic;">
    Integrate Angular with Node.js
  </h1>
  <h3>
    This is the Simple Web App with Express & Angular
  </h3>
  <h5>
    The below message is fetched from the express backend
  </h5>
  <p *ngIf="message">
    <b> {{ message.message }} </b>
  </p>
</div>
```

Step 11. Start the Development server angular fronted (ng server) and node.js backend (node server.js)

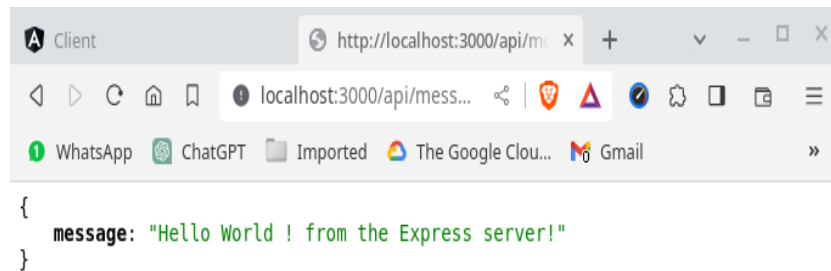
Output:

Angular (FrontEnd Application)



Full Stack Assignment

Node.js(BackEnd)



Q.8 Use MongoDB to store data in the Database.

Program:

```

require('dotenv').config(); // Load environment variables from .env
const express = require('express');
const mongoose = require('mongoose');

const app = express();
const port = process.env.PORT || 3000;
const dburl = process.env.dburl; // Get MongoDB connection URL from .env

async function checkMongoDBConnection() {
  try {
    await mongoose.connect(dburl, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    }); // Attempt to connect to MongoDB using Mongoose
    console.log('MongoDB connected successfully');
  } catch (error) {
    console.error('Error connecting to MongoDB:', error);
  }
}

// Initialize the Express server
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});

// Define the index type and index name
const loginSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true
  },

```

Full Stack Assignment

```
password:{
  type:String,
  required:true
}
});
//Store the record in the data ,data1
const collection = new mongoose.model('login',loginSchema);
data={
  name:"JavalaMukhi",
  password:"MCASEMESTER2"
}
data1={
  name:"Kaala Paani",
  password:"PTANHIKYAPASSWORDHAI"
}
//Insert the record into the Database
collection.insertMany([data]);
collection.insertMany([data1]);
// Check the MongoDB connection when the server starts
checkMongoDBConnection();
console.log('Data Inserted successully.');
```

Output:

```
_id: ObjectId('653fb6cc25756e2f6c676991')
name: "JavalaMukhi"
password: "MCASEMESTER2"
__v: 0
```

```
_id: ObjectId('653fb6cc25756e2f6c676992')
name: "Kaala Paani"
password: "PTANHIKYAPASSWORDHAI"
__v: 0
```