**Slip 8**

**Q1 Create a simple Angular application that fetches data from an API using**

**HttpClient. Implement an Observable to fetch data from an API endpoint.**

**app.module.ts:**

import { BrowserModule }

from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { HttpClientModule }

from '@angular/common/http';

import { AppRoutingModule }

from './app-routing.module';

import { AppComponent } from './app.component';

import { AddInputComponent }

from './add-input/add-input.component';

import { ShowApiComponent }

from './show-api/show-api.component';

@NgModule({

declarations: [

AppComponent,

ShowApiComponent

],

imports: [

BrowserModule,

AppRoutingModule,

HttpClientModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

**app.component.ts:**

import { Component, OnInit } from '@angular/core';

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

@Component({

selector: 'app-show-api',

templateUrl: './show-api.component.html',

styleUrls: ['./show-api.component.css']

})

export class ShowApiComponent implements OnInit {

li: any;

lis = [];

constructor(private http: HttpClient) {

}

ngOnInit(): void {

this.http.get(

'http://...com')

.subscribe(Response =& gt; {

// If response comes hideloader() function is called

// to hide that loader

if (Response) {

hideloader();

}

console.log(Response)

this.li = Response;

this.lis = this.li.list;

});

function hideloader() {

document.getElementById('loading').style.display = 'none';

}

}

}

**app.component.html:**

<h1>Registered Employees</h1>

<div class="d-flex justify-content-center">

<div class="spinner-border" role="status">

<span class="sr-only" id="loading">

Loading...

</span>

</div>

</div>

<table class="table" id='tab'>

<thead>

<tr>

<th scope="col">Name</th>

<th scope="col">Position</th>

<th scope="col">Office</th>

<th scope="col">Salary</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let e of lis;">

<td>{{ e.name }}</td>

<td>{{ e.position }}</td>

<td>{{ e.office }}</td>

<td>{{ e.salary }}</td>

</tr>

</tbody>

</table>

**Q2 Develop an Express.js application that defines routes for Create, Update**

**operations on a resource (Employee).**

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

const PORT = 3000;

// Middleware for parsing JSON bodies

app.use(bodyParser.json());

// In-memory "database" to store users

let users = [];

// Create a new user (Create)

app.post('/emp', (req, res) => {

const { name, email } = req.body;

const newEmp= { id:emp.length + 1, name, email };

emp.push(newEmp);

res.status(201).json(newEmp);

});

// Read all users (Read)

app.get('/emp', (req, res) => {

res.json(emp);

});

// Update a user by ID (Update)

app.put('/emp /:id', (req, res) => {

const userId = parseInt(req.params.id, 10);

const userIndex = users.findIndex(u => u.id === userId);

if (userIndex === -1) {

return res.status(404).json({ message: 'User not found' });

}

const { name, email } = req.body;

emp[userIndex] = { id: userId, name, email };

res.json(users[userIndex]);

});

// Start the server

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});