**Slip 1**

**Q1 Write an AngularJS script for addition of two numbers using ng-init, ngmodel**

**& ng-bind. And also Demonstrate ng-show, ng-disabled, ng-click**

**directives on button component.**

<!DOCTYPE html>

<html>

<head>

<script src="C:\angular-1.8.0\angular-1.8.0\angular.js"></script>

</head>

<body ng-app ng-init="checked=true;check=true" >

Enter number1: <input type="number" ng-model="n1"><br/>

<br/>

Enter number2:<input type="number" ng-model="n2">

<br/><br/>

<button ng-click="add=(n1+n2)" ng-init="add=0">result </button><br/><br/>

addition of two numbers=

<span ng-bind="add"></span><br/><br/>

<label>Click me to make button disabled or enabled: <input type="checkbox" ngmodel="

checked"></label>

<button ng-disabled="checked" >Disable</button><br/><br/>

<label>Click me to show or hide button: <input type="checkbox" ngmodel="

check"></label>

<button ng-show="check">Show</button><br/>

</body>

</html>

**Q2 Create a Node.js application that reads data from multiple files**

**asynchronously using**

**promises and async/await.**

const fs = require('fs');

const readFilePromise = (fileName, encoding) => {

return new Promise((resolve, reject) => {

fs.readFile(fileName, encoding, (err, data) => {

if (err) {

return reject(err);

}

resolve(data);

});

});

}

readFilePromise('./input.txt', 'utf8')

.then(data => {

console.log(data);

})

.catch(err => {

console.log(err);

});

**Slip2**

**Q1 Write an AngularJS script to print details of bank (bank name, MICR code,**

**IFC code, address etc.) in tabular form using ng-repeat**

<!DOCTYPE html>

<html>

<script src= "C:\angular-1.8.0\angular-1.8.0\angular.js">

</script>

<style>

body {

margin: 2%;

font-size: 120%;

}

th,

td {

padding: 20px;

}

</style>

<body ng-app="myApp" ng-controller="ListController">

<h1>Bank Details</h1>

<table border=1>

<thead>

<tr>

<th>S.No</th>

<th>Name</th>

<th>MICR Code</th>

<th>IFSC Code</th>

<th>Address</th>

</tr>

</thead>

<tr ng-repeat="i in bank">

<td> {{i.sno}} </td>

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

<td> {{i.micr}} </td>

<td> {{i.ifsc}} </td>

<td> {{i.address}} </td>

</tr>

</table>

</body>

<script>

var app = angular.module('myApp', []);

app.controller(

'ListController', function($scope) {

$scope.bank = [{

sno: 1,

name: 'SBI',

micr: 'sbi123',

ifsc: 9876563454,

address: "satara",

}, {

sno: 2,

name: 'BOI',

micr: 'boi123',

ifsc: 7865452396,

address: "Pune",

}, {

sno: 3,

name: 'RBI',

micr: 'rbi123',

ifsc: 7865452316,

address: "kolhapur",

}, {

sno: 4,

name: 'BOM',

micr: 'bom123',

ifsc: 7765458921,

address: "goa",

}, {

sno: 5,

name: 'BOB',

micr: 'bob123',

ifsc: 7765458921,

address: "satara",

}

];

});

</script>

</html>

**Q2 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>

**Slip 3**

**Q1. Write an AngularJS script to display list of games stored in an array on**

**click of button using ng-click. And also Demonstrate ng-init, ng-bind directive**

**of AngularJS.**

<!DOCTYPE html>

<html>

<head>

<script src="C:\angular-1.8.0\angular-1.8.0\angular.js"></script>

</head>

<body ng-app="myApp">

<div ng-controller="myCtrl" >

<button ng-click="myFunc()">Display Games</button>

<ol>

<li ng-repeat="i in game" ng-bind="i"></li>

<ol>

</div>

<script>

angular.module('myApp', [])

.controller('myCtrl', ['$scope', function($scope) {

$scope.count = 0;

$scope.myFunc = function() {

$scope.game=['Cricket','vollyball','Basketball'];

};

}]);

</script>

</body>

</html>

**Q2. find a company with a workforce greater than 30 in the array. used find by**

**id method**

interface Company {

name: string;

desc: string;

workForce: number;

}

const companies: Company[] = [

{ name: "GeeksforGeeks", desc: "A Computer Science Portal.", workForce: 200 },

{ name: "Company 2", desc: "Description 1", workForce: 30 },

{ name: "Company 3", desc: "Description 2", workForce: 10 },

];

const matchedCompany = companies.find(company => company.workForce > 30);

console.log(matchedCompany);

**Slip 4**

**Q1 fetch the details using ng-repeat in AngularJS**

<!DOCTYPE html>

<html>

<head>

<script src=

"https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">

</script>

<title>

Fetching the details using the ng-repeat Directive

</title>

<style>

body {

text-align: center;

font-family: Arial, Helvetica, sans-serif;

}

table {

margin-left: auto;

margin-right: auto;

}

</style>

</head>

<body ng-app="myTable">

<h1 style="color:green">GeeksforGeeks</h1>

<h3>

Fetching the details using the ng-repeat Directive

</h3>

<table ng-controller="control" border="2">

<tr ng-repeat="x in records">

<td>{{x.Country}}</td>

<td>{{x.Capital}}</td>

</tr>

</table>

<script>

var app = angular.module("myTable", []);

app.controller("control", function ($scope) {

$scope.records = [

{

"Country": "India",

"Capital": "Delhi"

},

{

"Country": "America ",

"Capital": "Washington, D.C. "

},

{

"Country": "Germany",

"Capital": "Berlin"

},

{

"Country": "Japan",

"Capital": "Tokyo"

}

]

});

</script>

</body>

</html>

**Q2. Express.js application to include middleware for parsing request bodies**

**(e.g., JSON, form data) and validating input data**

const express = require('express');

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

const { body, validationResult } = require('express-validator');

const app = express();

const PORT = 3000;

// Middleware for parsing JSON and form data

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

// Sample route with input validation

app.post(

'/submit',

[

// Validation checks

body('username').isLength({ min: 3 }).withMessage('Username must be at least 3

characters long'),

body('email').isEmail().withMessage('Enter a valid email address'),

body('age').isInt({ min: 1 }).withMessage('Age must be a positive integer'),

],

(req, res) => {

// Check validation results

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ errors: errors.array() });

}

// If validation passes, process the request

const { username, email, age } = req.body;

res.status(200).json({ message: 'Data received successfully', data: { username,

email, age } });

}

);

// Default route

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

res.send('Welcome to the Express.js application!');

});

// Start the server

app.listen(PORT, () => {

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

});

**Slip 5**

**Q1 Create a simple Angular component that takes input data and displays it.**

**app.component.ts**

// app.component.ts

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

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

message = 'Dynamic message from parent component';

}

**app.component.html**

<app-display [data]="message"></app-display>

**display.component.css**

.data-container {

padding: 1rem;

border: 1px solid #ccc;

border-radius: 5px;

margin-top: 1rem;

}

**Q2. Implement a simple server using Node.js**

// Importing the http module

const http = require("http")

// Creating server

const server = http.createServer((req, res) => {

// Sending the response

res.write("This is the response from the server")

res.end();

})

// Server listening to port 3000

server.listen((3000), () => {

console.log("Server is Running");

})

**Slip 6**

**Q1 Develop an Express.js application that defines routes for Create and Read**

**operations**

**on a resource (products).**

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('/products', (req, res) => {

const { name, price} = req.body;

const newProducts = { id: users.length + 1, name, email };

users.push(newProducts);

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

});

// Read all users (Read)

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

res.json(products);

});

// Read a single user by ID (Read)

app.get('/ products /:id', (req, res) => {

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

const product= products.find(u => u.id === products);

if (!user) {

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

}

res.json(user);

});

// Start the server

app.listen(PORT, () => {

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

});

**Q2 find a company with a workforce greater than 30 in the array. used find by**

**id method**

interface Company {

name: string;

desc: string;

workForce: number;

}

const companies: Company[] = [

{ name: "GeeksforGeeks", desc: "A Computer Science Portal.", workForce: 200 },

{ name: "Company 2", desc: "Description 1", workForce: 30 },

{ name: "Company 3", desc: "Description 2", workForce: 10 },

];

const matchedCompany = companies.find(company => company.workForce > 30);

console.log(matchedCompany);

**Slip 7**

**Q1 Create a Node.js application that reads data from multiple files**

**asynchronously using**

**promises and async/await.**

const fs = require('fs');

const readFilePromise = (fileName, encoding) => {

return new Promise((resolve, reject) => {

fs.readFile(fileName, encoding, (err, data) => {

if (err) {

return reject(err);

}

resolve(data);

});

});

}

readFilePromise('./input.txt', 'utf8')

.then(data => {

console.log(data);

})

.catch(err => {

console.log(err);

});

**Q2 Develop an Express.js application that defines routes for Create and Read**

**operations**

**on a resource (User).**

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('/users', (req, res) => {

const { name, email } = req.body;

const newUser = { id: users.length + 1, name, email };

users.push(newUser);

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

});

// Read all users (Read)

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

res.json(users);

});

// Read a single user by ID (Read)

app.get('/users/:id', (req, res) => {

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

const user = users.find(u => u.id === userId);

if (!user) {

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

}

res.json(user);

});

// Start the server

app.listen(PORT, () => {

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

});

**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}`);

});

**Slip 9**

**Q1 find a company with a workforce greater than 30 in the array. used find by**

**id method**

interface Company {

name: string;

desc: string;

workForce: number;

}

const companies: Company[] = [

{ name: "GeeksforGeeks", desc: "A Computer Science Portal.", workForce: 200 },

{ name: "Company 2", desc: "Description 1", workForce: 30 },

{ name: "Company 3", desc: "Description 2", workForce: 10 },

];

const matchedCompany = companies.find(company => company.workForce > 30);

console.log(matchedCompany);

**Q2 Create Express.js application to include middleware for parsing request**

**bodies (e.g., JSON, form data) and validating input data. Send appropriate**

**JSON**

**responses for success and error cases.**

const express = require('express');

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

const { body, validationResult } = require('express-validator');

const app = express();

const PORT = 3000;

// Middleware for parsing JSON and form data

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

// Sample route with input validation

app.post(

'/submit',

[

// Validation checks

body('username').isLength({ min: 3 }).withMessage('Username must be at least 3

characters long'),

body('email').isEmail().withMessage('Enter a valid email address'),

body('age').isInt({ min: 1 }).withMessage('Age must be a positive integer'),

],

(req, res) => {

// Check validation results

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ success: false, errors: errors.array() });

}

// If validation passes, process the request

const { username, email, age } = req.body;

res.status(200).json({ success: true, message: 'Data received successfully', data: {

username, email, age } });

}

);

// Default route

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

res.send('Welcome to the Express validation application!');

});

// Start the server

app.listen(PORT, () => {

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

});

**Slip 10**

**Q1 Implement a simple server using Node.js**

const http = require("http")

// Creating server

const server = http.createServer((req, res) => {

// Sending the response

res.write("This is the response from the server")

res.end();

})

// Server listening to port 3000

server.listen((3000), () => {

console.log("Server is Running");

})

**Q2 Extend the previous Express.js application to include middleware for**

**parsing request**

**bodies (e.g., JSON, form data) and validating input data. Send appropriate**

**JSON**

**responses for success and error cases.**

const express = require('express');

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

const { body, validationResult } = require('express-validator');

const app = express();

const PORT = 3000;

// Middleware for parsing JSON and form data

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

// Sample route with input validation

app.post(

'/submit',

[

// Validation checks

body('username').isLength({ min: 3 }).withMessage('Username must be at least 3

characters long'),

body('email').isEmail().withMessage('Enter a valid email address'),

body('age').isInt({ min: 1 }).withMessage('Age must be a positive integer'),

],

(req, res) => {

// Check validation results

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ success: false, errors: errors.array() });

}

// If validation passes, process the request

const { username, email, age } = req.body;

res.status(200).json({ success: true, message: 'Data received successfully', data: {

username, email, age } });

}

);

// Default route

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

res.send('Welcome to the Express validation application!');

});

// Start the server

app.listen(PORT, () => {

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

});

**Slip 11**

**Q1 Develop an Express.js application that defines routes for Create operations**

**on a resource (Movie).**

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 movies= [];

// Create a new user (Create)

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

const { name, actro} = req.body;

const newmovies = { id: movies.length + 1, name, email };

movies.push(newmovies);

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

});

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

res.json(movies);

});

// Start the server

app.listen(PORT, () => {

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

});

**Q2 Create Angular application that print the name of students who play**

**basketball using filter and map method**

// Taking an array of Student object

let students = [

{ id: "001", name: "Anish", sports: "Cricket" },

{ id: "002", name: "Smriti", sports: "Basketball" },

{ id: "003", name: "Rahul", sports: "Cricket" },

{ id: "004", name: "Bakul", sports: "Basketball" },

{ id: "005", name: "Nikita", sports: "Hockey" }

]

let basketballPlayers = students.filter(function (student) {

return student.sports === "Basketball";

}).map(function (student) {

return student.name;

})

console.log("Basketball Players are:");

// Printing out the name of Basketball players

basketballPlayers.forEach(function (players) {

console.log(players);

});

**Slip 12**

**Q1Write an AngularJS script to print details of Employee (employee name,**

**employee Id, address etc.) in tabular form using ng-repeat**

<!DOCTYPE html>

<html>

<script src= "C:\angular-1.8.0\angular-1.8.0\angular.js">

</script>

<style>

body {

margin: 2%;

font-size: 120%;

}

th,

td {

padding: 20px;

}

</style>

<body ng-app="myApp" ng-controller="ListController">

<h1>Bank Details</h1>

<table border=1>

<thead>

<tr>

<th>S.No</th>

<th>Name</th>

<th>Emp Id </th>

<th>Address</th>

</tr>

</thead>

<tr ng-repeat="i in bank">

<td> {{i.sno}} </td>

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

<td> {{i.micr}} </td>

<td> {{i.ifsc}} </td>

<td> {{i.address}} </td>

</tr>

</table>

</body>

<script>

var app = angular.module('myApp', []);

app.controller(

'ListController', function($scope) {

$scope.bank = [{

sno: 1,

name: 'SBI',

micr: 'sbi123',

address: "satara",

}, {

sno: 2,

name: 'BOI',

micr: 'boi123',

address: "Pune",

}, {

sno: 3,

name: 'RBI',

micr: 'rbi123',

address: "kolhapur",

}, {

sno: 4,

name: 'BOM',

micr: 'bom123',

address: "goa",

}, {

sno: 5,

name: 'BOB',

micr: 'bob123',

address: "satara",

}

];

});

</script>

</html>

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

**on a resource (User).**

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 **User** = [];

// Create a new user (Create)

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

const { name, actro} = req.body;

const new**User** = { id: movies.length + 1, name, email };

movies.push(new**User**);

res.status(201).json(new**User**);

});

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

res.json(**User**);

});

// Start the server

app.listen(PORT, () => {

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

});

**Slip 13**

**Q1 Extend the previous Express.js application to include middleware for**

**parsing request**

**bodies (e.g., JSON, form data) and validating input data. Send appropriate**

**JSON**

**responses for success and error cases.**

const express = require('express');

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

const { body, validationResult } = require('express-validator');

const app = express();

const PORT = 3000;

// Middleware for parsing JSON and form data

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

// Sample route with input validation

app.post(

'/submit',

[

// Validation checks

body('username').isLength({ min: 3 }).withMessage('Username must be at least 3

characters long'),

body('email').isEmail().withMessage('Enter a valid email address'),

body('age').isInt({ min: 1 }).withMessage('Age must be a positive integer'),

],

(req, res) => {

// Check validation results

const errors = validationResult(req);

if (!errors.isEmpty()) {

return res.status(400).json({ success: false, errors: errors.array() });

}

// If validation passes, process the request

const { username, email, age } = req.body;

res.status(200).json({ success: true, message: 'Data received successfully', data: {

username, email, age } });

}

);

// Default route

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

res.send('Welcome to the Express validation application!');

});

// Start the server

app.listen(PORT, () => {

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

});

**Q2 Create a simple Angular component that takes input data and displays it.**

**app.component.ts**

// app.component.ts

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

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

message = 'Dynamic message from parent component';

}

**app.component.html**

<app-display [data]="message"></app-display>

**display.component.css**

.data-container {

padding: 1rem;

border: 1px solid #ccc;

border-radius: 5px;

margin-top: 1rem;

}

**Slip 14**

**Q1 Create Angular application that print the name of students who got 85%**

**using filter and map method**

let students = [

{ id: "001", name: "Anish",Percentage: "40%" },

{ id: "002", name: "Smriti", Percentage: "25%" },

{ id: "003", name: "Rahul", Percentage: "85%" },

{ id: "004", name: "Bakul", Percentage: "96%" },

{ id: "005", name: "Nikita", Percentage: "90%" }

]

let basketballPlayers = students.filter(function (student) {

return student.Percentage === "85%";

}).map(function (student) {

return student.name;

})

console.log("Basketball Players are:");

// Printing out the name of Basketball players

basketballPlayers.forEach(function (players) {

console.log(players);

});

**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}`);

});

**Slip 15**

**Q1. find a Emp with a Salary greater than 25000 in the array. used find by**

**id method**

interface Company {

name: string;

desc: string;

workForce: number;

}

const companies: Company[] = [

{ name: "GeeksforGeeks", desc: "A Computer Science Portal.", Salary: 25000 },

{ name: "Company 2", desc: "Description 1", Salary: 30000 },

{ name: "Company 3", desc: "Description 2", Salary: 10000 },

];

const matchedCompany = companies.find(company => company. Salary > 25000);

console.log(matchedCompany);

**Q2 Create Angular application that print the name of students who got 85%**

**using filter and map method**

let students = [

{ id: "001", name: "Anish",Percentage: "40%" },

{ id: "002", name: "Smriti", Percentage: "25%" },

{ id: "003", name: "Rahul", Percentage: "85%" },

{ id: "004", name: "Bakul", Percentage: "96%" },

{ id: "005", name: "Nikita", Percentage: "90%" }

]

let basketballPlayers = students.filter(function (student) {

return student.Percentage === "85%";

}).map(function (student) {

return student.name;

})

console.log("Basketball Players are:");

// Printing out the name of Basketball players

basketballPlayers.forEach(function (players) {

console.log(players);

});