SDAC
Making Nation Employable

Name. ______________________________ Batch Code: ________

Scripting using JavaScript (JS) & TypeScript (TS)

DAY (1 to 9)	Topic	Actual Date	Planned Date	Status (Completed)	Comment
1	Introduction To Software				
	Introduction To Scripting Language				
	Introduction To JavaScript				
	ECMAScript				
	Data Type				
	Variable (Scope & Variable Initialization)				
	Control Flow & loops				
	Functions				
	Types of functions(Arrow Function, Anonymous Functions ,Named Functions)				
	Arrays (Methods like push, pop, shift, unshift, map, filter, and reduce).				
	Object (Creating and accessing object properties, object methods).				
2	Practical				
3	Es6+ Features (template literal, destructuring, class, rest parameter, default parameter)				
	Asynchronous JavaScript (callback, promise, async /await)				
	Error Handling				
	Module (Import /Export)				
4	Practical				
5	Dom Manipulation				
	Events & Event Handling				
	Client Side Validation (local storage)				
6	Practical				
7	Introduction to Typescript Advantages of TypeScript over JavaScript (static typing, improved maintainability).				
	Data Types				
	Variables (Variable Declaration ,Variable Initialization)				
	Class , Constructor & Method				
	Object				
	Abstract Class				
	Interface				
	Function And Generics				
8	Practical				
9	Integrating MySQL with JavaScript & Node.js				
10	Final Project				

I acknowledge that I am fully satisfied with the session and have gained a clear understanding of all the topics covered.

Sign: ______________

CLASSWORK 1

No.	Practical Questions	Status (Completed)
1	Create an object called person with properties name, age, and gender. Access and log the name property to the console.	
2	Create an arrow function named square that takes a number as a parameter and returns the square of that number.	
3	Implement an anonymous function assigned to the variable multiplyByTwo that Take number as a parameter and multiplies each element by 2.	
4	Write a named function filterEvenNumbers that takes an array of numbers as a parameter and returns a new array containing only the even numbers.	
5	Demonstrate array method (map, filter, reduce) a. Map to square each number. b. Filter to extract odd number. c. Reduce to calculate the sum of all elements.	
6	Create a Function findMax That Accepts an Array of Numbers and Returns the Largest Number in the Array.	
7	Write a Function to Filter Employees Earning More Than a Certain Amount from an Array of Employee Objects.	
8	Write a program to find the total sales from an array of Order objects in an e- commerce system.	

HOMEWORK 1

No.	Practical Questions	Status (Completed)
1	Create an object called student with properties: name, grade, and subject. Access and log the grade property to the console.	
2	Create an arrow function named calculateGrade that takes a score as a parameter and returns the grade (e.g., A, B, C).	
3	Implement an anonymous function assigned to the variable increaseMarks that takes an array of marks as a parameter and increases each mark by 10.	
4	Write a named function filterPassedStudents that takes an array of student marks and returns a new array containing only the students who passed.	
5	Demonstrate array methods (map, filter, reduce): a. Use map to convert marks to grades. b. Use filter to extract students who scored above 75. c. Use reduce to calculate the total marks of all students.	
6	Create a function findTopper that accepts an array of student marks and returns the highest mark.	
7	Write a function to filter students with marks greater than 80 from an array of student objects.	
8	Write a program to find the total marks scored by all students in a class.	

CLASSWORK 2

No.	Practical Questions	Status (Completed)
1	Provide an example of array destructuring and object destructuring in JavaScript.	
2	Create a class named Emp with following attribute name, id, salary & displayInfo method provide access code for the same.	
3	Demonstrate an example for Promise.	
4	Demonstrate an example of Async and await.	
5	Demonstrate an example for default parameter and rest parameter.	
6	Show Uses of Try /Catches	
7	Show an Example of Exception Propagation	
8	Show an Example of Finally Block	
9	Create a User Defined exception.	
10	Show example of Throw.	

HOMEWORK 2

No.	Practical Questions	Status (Completed)
1	Provide an example of array destructuring and object destructuring in JavaScript to extract product names from an array of products and their details from a product object in an online shopping system.	
2	Create a class named Product with the following attributes: name, productId, price, and a displayDetails method. Provide code to access and display these details for a product.	
3	Demonstrate an example of a Promise that simulates fetching product details (e.g., name, price, availability) from an online store's server.	
4	Demonstrate an example of async and await to simulate fetching and displaying a customer's order history from an online shopping system.	
5	Demonstrate an example for default parameters and rest parameters in a function that calculates the total cost of items in a customer's shopping cart. Use a default parameter for applying a discount rate and rest parameters for the list of product prices.	
6	Show the uses of try/catch when processing a payment during the checkout process, handling possible errors like payment failure or network issues.	
7	Show an example of exception propagation when adding an item to a shopping cart if the product is out of stock in the online store's inventory.	
8	Show an example of a finally block that executes when updating customer profile information, regardless of whether the operation was successful or resulted in an error.	
9	Create a user-defined exception called InvalidCouponCode and demonstrate how it is used when a customer tries to apply an invalid coupon code during checkout.	
10	Show an example of throw where a function throws an exception if a product's quantity in the inventory falls below a certain threshold (e.g., less than 5 units)	

CLASSWORK 3

No.	Practical Questions	Status (Completed)
1	Demonstrate how to create a new <li> element, set its text content, and append it to an existing <ul> element on a web page using JavaScript. Add functionality to store the text content of the new <li> element in local storage.	
2	Implement an event listener that triggers an alert when a button with the ID "myButton" is clicked on a web page.	
3	Create a new button "clearButton" that, when clicked, clears the click count from local storage and resets the counter.	
4	Create a form with fields for name, email, and password. Using JavaScript, implement client-side validation for the following criteria: a. Name: Should contain only alphabetic characters. b. Email: Should match a valid email format. c. Password: Should have a minimum length of 8 characters, including at least one uppercase letter, one lowercase letter, and one digit. d. After validating the form, store the name and email in local storage	
5	Create a contact form with fields for name, email, and query. Implement JavaScript validation to ensure: a. Name: Is not empty and contains only alphabetic characters. b. Email: Matches a valid email format. c. Query: Is not empty and less than 1000 characters. d. After validation, store the name, email, and query in local storage	

HOMEWORK 3

No.	Practical Questions	Status (Completed)
1	Write JavaScript to create a new <li> element representing a product name, set its text content, and append it to an existing <ul> of products. Store the product name in local storage	
2	Implement an event listener that triggers an alert showing the number of items added to the cart when a button with the ID "addToCart" is clicked. Track the item count in local storage.	
3	Create a "resetCart" button to clear the cart count from local storage.	
4	Create a product form with fields for product name, price, and quantity. Use JavaScript for client-side validation: a. Product Name: Should not be empty. b. Price: Should be a positive number. c. Quantity: Should be a numeric value greater than 0. d. Store the product's details in local storage upon successful validation.	
5	Create an order form with fields for customer name, product, and address. Implement JavaScript validation to ensure: a. Customer Name: Is not empty and contains only alphabetic characters. b. Product: Matches one of the available products. c. Address: Is not empty. d. Store each order (customer name, product, address) in local storage after validation.	

CLASSWORK 4

No.	Practical Questions	Status (Completed)
1	Create class Emp with its important attributes like a. Name/id/salary/address b. Method – display Info	
2	Create a class which contains static and non-static members.	
3	Show an example for parameterized constructor.	
4	Demonstrate an example for abstract class.	
5	Demonstrate an example for Interface.	
6	Demonstrate example for generic in function.	
7	Demonstrate real time use of abstract class and interface.	

HOMEWORK 4-TS

No	Practical Questions	Status (Completed)
1	Define a class named Student with the following attributes: a. name, studentId, grade, address. b. Include a method displayInfo() that prints all the student's details.	
2	Create a class School that contains: a. Static member: totalStudents to track the total number of students. b. Non-static member: studentList to store individual student details. c. Methods to increment the static member and add student details to the non-static list.	
3	Create a Course class with a parameterized constructor that takes courseName, courseCode, and instructor as parameters and assigns them to class attributes.	
4	Define an abstract class Person with abstract methods getDetails() and getRole(). Create two derived classes Student and Teacher that implement these methods to return their respective details and roles.	
5	Create an interface Attendance with a method markAttendance(). Implement this interface in the Student class, and define how attendance is marked for a student.	
6	Create a generic function getStudentInfo<T>(info: T): T that returns any type of student information, such as name, ID, or grade.	
7	Define an abstract class Institute with an abstract method getInstitutionType(). Create a class School implementing the interface Management with methods like addStudent(), removeStudent(). Combine these in a real-time use case to manage different types of institutions like School and College.	

CLASSWORK 5

No.	Practical Questions	Status (Completed)
1	Create a MySQL database named school and a table named students with fields: 1. id (INT AUTO_INCREMENT PRIMARY KEY) 2. name (VARCHAR) 3. grade (VARCHAR) 4. subject (VARCHAR)	
2	Perform the following CRUD Operation: 1. Insert 2. Update 3. Delete 4. Show	

HOMEWORK 5

No.	Practical Questions	Status (Completed)
1	Create a MySQL database named emp and a table named students with fields: 1. id (INT AUTO_INCREMENT PRIMARY KEY) 2. name (VARCHAR) 3. salary (INT) 4. Department (VARCHAR)	
2	Perform the following CRUD Operation: 1. Insert 2. Update 3. Delete 4. Show	