18. Working with TypeScript and React Native features

Aim/Objective: The objective of this experiment is to be familiar with Typescript and React Native

Description:

TypeScript is a programming language that is a superset of JavaScript. It adds optional static typing to JavaScript, enabling developers to catch errors and provide type safety during development. TypeScript code is transpiled into JavaScript, which can be executed in any JavaScript runtime environment.

React Native is an open-source framework for building cross-platform mobile applications. It allows developers to write mobile apps using JavaScript and the React framework, which is popular for building user interfaces for web applications.

Pre-Requisites:

Node js, Web Browsers, express, Postman, nodemon.

>npm install tsc typescript

Pre-Lab:

1. **What is TypeScript, and how does it differ from JavaScript?**

   TypeScript is a statically typed superset of JavaScript, adding optional type annotations and other features while transpiling to plain JavaScript for execution.

2. **Explain the benefits of using TypeScript in a React Native project.**

   Using TypeScript in a React Native project provides benefits such as improved code quality, enhanced developer tooling, early error detection, and better code maintainability through static typing and improved IDE support.

3. **What is static typing, and how does TypeScript leverage it for type safety?**

   Static typing is a programming language feature that enforces data types at compile-time, and TypeScript leverages it by allowing developers to specify and check types for variables, function parameters, and return values, enabling early detection of type-related errors and improving code reliability.

4. **What is React Native?**

   React Native is an open-source framework for building mobile applications using JavaScript and React, allowing developers to create cross-platform apps that run on both iOS and Android with a single codebase.

5. **How does TypeScript's type inference feature work?**

   TypeScript's type inference automatically deduces the data types of variables and expressions based on their usage, helping developers write statically typed code with less manual type annotations.

In-Lab:

Exercise 1: Write a TypeScript code for addition of 2 numbers. Transpile the the code into Javascipt by using tsc. Execute the transpiled code in Node environment.

Procedure/Program:

Step 1: Install TypeScript

If you haven't already installed TypeScript, follow these steps:

1. Open your command prompt or terminal.
2. Run the following command to install TypeScript globally using npm:

Command: npm install -g typescript

Step 2: Create a TypeScript file

Create a TypeScript file named **addition.ts** using a text editor of your choice. You can use Notepad, Visual Studio Code, or any other code editor.

Command: notepad addition.ts

Step 3: Write TypeScript Code

In **addition.ts**, add the following TypeScript code for adding two numbers:

```
function addNumbers(a: number, b: number): number {
  return a + b;
}

const num1: number = 5;
const num2: number = 7;
const sum: number = addNumbers(num1, num2);

console.log(`The sum of ${num1} and ${num2} is ${sum}`);
```

Step 4: Transpile TypeScript to JavaScript

Open your command prompt or terminal and navigate to the directory where your **addition.ts** file is located.

Use the TypeScript compiler (tsc) to transpile the TypeScript code into JavaScript. Run the following command

Command: tsc addition.ts

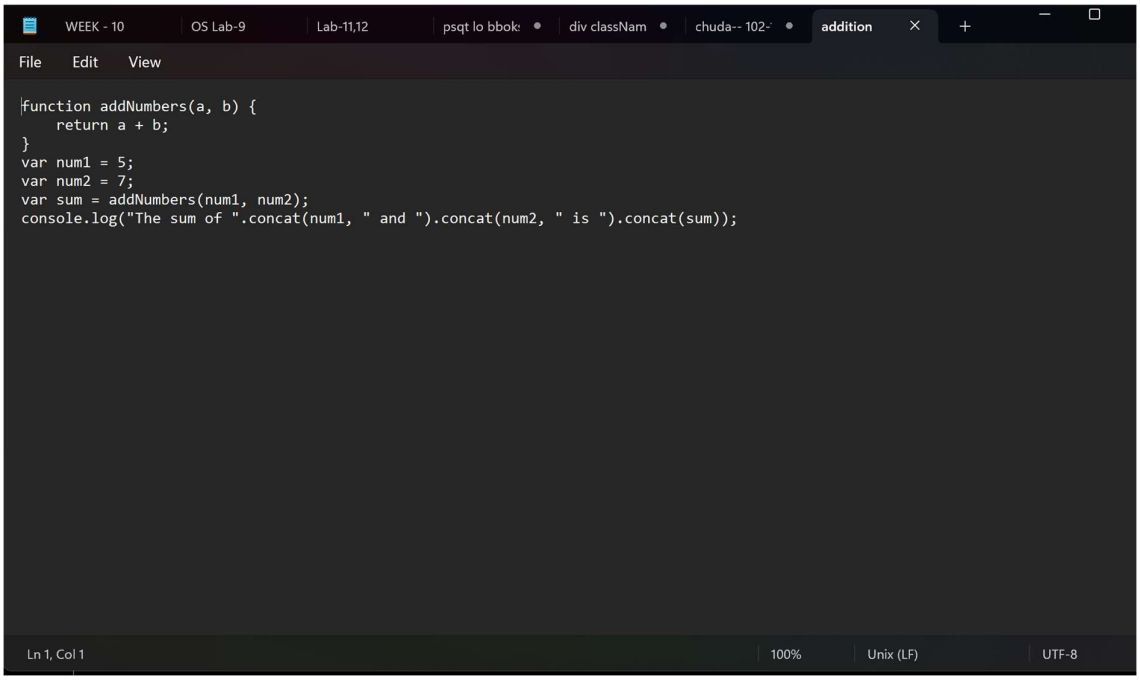
This will generate a JavaScript file named **addition.js** in the same directory.

Step 5: Execute Transpiled JavaScript

Now that you have the transpiled JavaScript file, you can execute it using Node.js. Run the following command in your terminal:

Command: node addition.js

- **Data and Results:**



Output:

C:\Windows\System32>npm install -g typescript

changed 1 package in 2s

C:\Windows\System32>notepad addition.js

C:\Windows\System32>tsc addition.ts

C:\Windows\System32>node addition.js

The sum of 5 and 7 is 12

C:\Windows\System32>

Sample VIVA-VOCE Questions (In-Lab):

1. **How does TypeScript's type inference feature work?**
2. **Describe how TypeScript interfaces are used in React Native development.**
3. **What are the advantages of using TypeScript in terms of code maintainability and collaboration in a team?**
4. **How can TypeScript help catch potential errors during development in a React Native application?**
5. **How does React Native leverage native components and APIs for performance optimization?**

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date