Week 2 - Assignment

Weekly Assignment Covering HTML/CSS/Tailwind and JavaScriptBasics

Week 2 – Weekly Assignment covering HTML, CSS, Tailwind and Basic Java Script

Assignment 1:

You are working as a Frontend Developer for an Insurance Company. Your task is to build an Enquiry & Feedback Form for customers to submit policy-related queries and feedback.

🧩 Section A – HTML Structure (estimated time 30 mins)

- 🔹 Task 1: Create Page Layout: Create a basic HTML page with:
  - Page title: Insurance Enquiry & Feedback
  - A centered form container
  - Heading and short description
- 🔹 Task 2: Form Fields: Create a form with the following fields:

Field	Type	Required
Full Name	Text input	Yes
Email	Email input	Yes
Mobile Number	Tel input	Yes
Request Type	Dropdown	Yes
Policy Type	Dropdown	Yes
Message	Textarea	Yes
Experience Rating	Radio buttons (1–5)	Yes
Submit Button	Button	—

- 🔹 Task 3: Dropdown Values

  Request Type
  - Policy Enquiry
  - Claim Related
  - Feedback

  Policy Type
  - Health Insurance
  - Life Insurance
  - Vehicle Insurance
  - Home Insurance

🎨 Section B – CSS Styling (30 mins)

- 🔹 Task 4: Form Styling

Apply CSS to achieve:

- Centered form with max-width
- White background card
- Rounded corners
- Box shadow
- Consistent spacing between fields

- 🔹 Task 5: Input & Button Design
  - Inputs and textarea should be full width
  - Add padding and border radius
  - Button should have:
    - Insurance brand color (blue)

- Hover effect

◆ Task 6: Error Styling

- Display error messages below each field
- Error text color should be red
- Use small font size

⚙️ Section C – JavaScript Validation (30 mins)

◆ Task 7: Form Validation Rules

On form submit:

- Prevent default submission
- Validate:
  - Name → not empty
  - Email → not empty
  - Mobile → exactly 10 digits
  - Request Type → selected
  - Policy Type → selected
  - Message → minimum 10 characters
  - Rating → must be selected

◆ Task 8: Error Display

- Show error message under respective field
- Clear old error messages before validation
- Do NOT show browser default alerts

◆ Task 9: Success Message

If all validations pass:

- Show a green success message:
  "Thank you! Your enquiry has been successfully submitted."
- Reset the form

❖❖

Assignment 2:

You are required to build a Customer Dashboard for an Insurance Company. The dashboard will help insurance agents view, filter, and manage customer insurance enquiries.

Customers can:

- View available insurance plans
- Submit enquiry forms
- See policy details dynamically

Agents can:

- View customer list
- Filter customers by policy type
- Calculate premium dynamically

🧩 What You Will Build

1️⃣ Landing Section – Insurance Overview

- Company name & tagline
- Hero section with CTA button
- Statistics cards (Customers, Policies, Claims)

2️⃣ Insurance Plans Section

- Health Insurance
- Life Insurance
- Vehicle Insurance
- Each card shows:
  - Plan Name
  - Base Premium
  - Coverage Amount
  - **Enroll button**

3️⃣ **Customer Enquiry Form**

- Customer Name
- Age
- Email
- Policy Type (dropdown)
- Coverage Amount (range slider)
- Submit button
- Form validation using JavaScript

4️⃣ **Customer List Table (Dynamic)**

- List of customers added via form
- Columns:
  - Name
  - Age
  - Policy Type
  - Coverage
  - Estimated Premium
- Filter customers by policy type
- Search customer by name

🛠️ **Technical Requirements**

✅ **HTML**

- Semantic structure (header, section, article, footer)
- Accessible form fields
- Table for customer listing

✅ **CSS (Basic)**

- Custom CSS for:
  - Form focus states
  - Button hover effects
  - Error messages

✅ **Tailwind CSS (Major Styling)**

Must use:

- Flexbox & Grid
- Responsive breakpoints
- Utility classes for spacing, typography, colors
- Cards, buttons, badges

🎯 **Consider Following:**

1. Customer Object Structure

```
{
    id: number,
```

```
    name: string,
    age: number,
    policyType: string,
    coverage: number,
    premium: number
}
```

2. 2. Dynamic Premium Calculation
   Rules:
   - Base premium depends on policy type
   - Extra premium if age > 45
   - Coverage amount affects final premium

   Example
   Health → ₹3000 base
   Life → ₹5000 base
   Vehicle → ₹2000 base

   +20% if age > 45
   +₹500 per additional 1L coverage

3. 🧪 **Functional Tasks (What Student Must Implement)**
   - **Task 1: Display Insurance Plans**
     Render insurance plans using JavaScript array
     Use map() to generate cards dynamically
   - **Task 2: Handle Form Submission**
     Prevent default submit
     Validate inputs
     Create customer object
     Push to customers array
     Render table
   - **Task 3: Calculate Premium**
     Write a reusable function
     calculatePremium(age, policyType, coverage)
   - **Task 4: Render Customer Table**
     Use map() to render rows
     Update table dynamically without page reload
   - **Task 5: Filter Customers**
     Dropdown to filter by policy type
     Use filter() to update table
   - **Task 6: Search Customer**
     Search by name (case-insensitive)
     Combine filter + search
   - **Task 7: Dashboard Summary**
     Show:
     Total customers
     Total premium value

Use reduce() for calculations

📱 **Responsiveness Requirements**

Mobile: stacked layout

Tablet: 2-column grid

Desktop: full dashboard layout

⏱️ **Ideal Time Breakdown**

- Layout & Tailwind styling → 35–40 mins
- JavaScript logic → 35–45 mins
- Filtering & search → 15 mins
- Debugging & polish → 10–15 mins

❖❖❖

Assignment 3:

Given here is a Debugging Exercise – "Find the Bug" version based on the same Insurance Customer JavaScript module we covered in class.

Insurance Customer Analytics – Find the Bug

⏱️ **Estimated Duration : 45–60 minutes**

🎯 **Objective**

Identify and fix **logical, syntax, and ES6 usage bugs** related to:

- let / const
- Arrow functions
- Template literals
- map, filter, reduce
- Loops
- Immutability

📦 **Given Data (Correct – Do NOT Change)**

```
const customers = [
  { id: 1, name: "Ravi", age: 32, policy: "Health", premium: 4800, active: true },
  { id: 2, name: "Anita", age: 51, policy: "Life", premium: 12000, active: true },
  { id: 3, name: "Kiran", age: 28, policy: "Vehicle", premium: 3500, active: false },
  { id: 4, name: "Meena", age: 45, policy: "Health", premium: 6000, active: true },
  { id: 5, name: "Suresh", age: 60, policy: "Life", premium: 18000, active: false }
];
```

🧩 **Instructions**

- Each snippet **contains at least one bug**
- Identify:
    - ❌ What is wrong
    - ✅ Corrected code
- Do **not rewrite completely** – fix only the bug

🐞 **Bug 1: Loop Output Issue**

```
for (let i = 0; i <= customers.length; i++) {
  console.log(customers[i].name);
}
```

🔍 **Hint:** Array index & loop condition

🐞 Bug 2: filter() Not Working

```
const activeCustomers = customers.filter((c) => {
  c.active === true;
});
```

🔍 **Hint:** Return value

🐞 Bug 3: Premium Increase Logic Broken

```
const updatedPremiums = customers.map((c) => {
  if (c.age >= 50) {
    c.premium = c.premium * 1.1;
  }
});
```

🔍 **Hint:** map return & immutability

🐞 Bug 4: Wrong Total Premium Calculation

```
const totalPremium = customers.reduce((total, c) => {
  total + c.premium;
}, 0);
```

🔍 **Hint:** reduce return

🐞 Bug 5: Template Literal Not Printing

```
console.log("Customer ${customers[0].name} has policy
${customers[0].policy}");
```

🔍 **Hint:** Quotes type

🐞 Bug 6: Policy Count Incorrect

```
const policyCount = customers.reduce((count, c) => {
  count.policy = (count.policy || 0) + 1;
  return count;
}, {});
```

🔍 **Hint:** Dynamic object key

🐞 Bug 7: Risk Level Always Undefined

```
const customersWithRisk = customers.map((c) => {
  let riskLevel;
  if (c.age < 35) riskLevel = "Low";
  if (c.age <= 50) riskLevel = "Medium";
  else riskLevel = "High";
  return { ...c, riskLevel };
});
```

🔍 **Hint:** Condition chaining

🐞 Bug 8: Active vs Inactive Count Wrong

```
let active = 0,
  inactive = 0;

for (const c in customers) {
  if (c.active) active++;
  else inactive++;
}
```

🔍 **Hint:** for...in vs for...of

🐞 Bug 9: Arrow Function Syntax Error

```
const getLifeCustomers = () =>
  customers.filter((c) => c.policy === "Life").map((c) =>
c.name);
```

🔍 **Hint:** Arrow function return

🐞 Bug 10: Sorting Mutates Original Array

```
const sortedCustomers = customers.sort((a, b) => b.premium -
a.premium);
```

🔍 **Hint:** Array mutation

Good Luck!!