National University of Computer & Emerging Sciences

Department of Software Engineering

Course: Web Engineering - Spring 2025

Assignment 01

Deadline: 13 February, 2025
Total Marks: 55

Instructions:

1. This assignment consists of **5 scenario-based questions (10 marks each)** and **1 bonus mark in each question**.
2. You must provide complete and functional HTML, CSS, and JavaScript code for each question.
3. Submissions must be in a ZIP file containing your HTML, CSS, and JS files.
4. For some questions, your implementation must be uniquely based on your roll number (format: 22F-XXXX), ensuring different logic for each student.
5. The assignments will be checked through **MOSS**, so avoid plagiarism among each other.
6. Submit your assignment as a single ZIP file named 22F-XXXX_Web_Assignment_1.zip
7. Submissions not following the submission structure of the assignment will receive **deductions.**
8. Submissions **must follow the given file structure:**

```
22F-XXXX_Web_Assignment_1/
├── Q1/
|  ├── index.html
|  ├── style.css
|  ├── script.js
|  ├── README.md (Write your explanation in the first person:  "I implemented this by..." )
├── Q2/
|  ├── index.html
|  ├── script.js
|  ├── README.md (Write your explanation in the first person:  "I approached this by..." )
... (repeat for all 5 questions)
```

9. Directly copying from peers/online sources/AI will result in 80% marks deductions in the assignment for both students.

Task 1 – Unique Product Card Design:

Objective:

Design and implement a visually unique product card using only CSS and JavaScript. The product card should feature a **3D flip animation**, where the front side displays basic product details, and the back side reveals additional information when hovered.

Requirements:

- The product card must have a front and back side. When hovered, the card should perform a smooth 3D flip to reveal the back side. The transition should be animated using **only CSS** (no JavaScript-based animations).
- The card content must be structured using **CSS Grid**. There should be proper alignment of text, images, and buttons within the grid.
- Product details should be fetched from **a JavaScript object** instead of hardcoded HTML. The object should contain **multiple products**, and the JavaScript code should dynamically create the product cards.
- The front side should display product **name**, **price**, and **image**. The back side should display a short product description.

Constraints:

- **No frameworks** like Bootstrap, Tailwind, or jQuery.

Bonus:

- Add a "**Buy Now**" button that logs the selected product's name to the console when clicked.
- Implement a **dark mode** toggle for the product cards.

Task 2 – Interactive To-Do List with Priority Management:

Objective:

Develop an **interactive to-do list** where users can **add, delete, and prioritize tasks** dynamically. The tasks should be categorized by priority levels, and users should be able to mark tasks as **completed**, moving them to a separate section. The implementation should use **arrays, map(), reduce(), event delegation, and DOM manipulation**.

Requirements:

- Users can add tasks with a **name** and a **priority level** (**High, Medium, Low**).
- Tasks should be displayed dynamically using **map()** and grouped by priority.
- Each task should have a **"Complete" button** to move it to the completed list.
- Clicking **"Delete"** should remove a task from the list.
- Use **reduce()** to display the **total number of incomplete tasks**.

Constraints:

- **Vanilla JavaScript only** (no external libraries).
- Use **event delegation** for button clicks instead of multiple event listeners.
- The app should dynamically update the UI without requiring a page refresh.
- Prevent adding **empty tasks** and ensure UI updates correctly when tasks are completed or deleted.

Bonus:

- **Search & Filter** to find tasks by name or priority.
- **Dark Mode** toggle.

Task 3 – Selective String Reversal with Dynamic List Population

Objective:

Develop an interactive web-based tool that processes a user-inputted string by reversing it while skipping characters at specific intervals. The skip interval is determined by summing the digits of **your roll number**. The transformed result should be displayed dynamically on the webpage.

Requirements:

- Create an **HTML form** with fields for a string and a roll number, plus a "Transform" button.
- Extract digits from the roll number and compute their sum to determine the **skip interval (N)**.
- Reverse the string while skipping every **N-th character**, preserving their original positions.
- Display the original and transformed strings in a **dynamically updated list** using **DOM manipulation**.
- Implement logic using **arrow functions, map(), reduce(), sets, and JavaScript functions**.

Constraints:

- Handle edge cases like **N > string length.**
- Maintain correct positioning of **spaces.**
- Use **only vanilla JavaScript, HTML, and CSS.**

Bonus:

- Allow users to manually enter a skip interval instead of using the **roll number sum**.

Task 4 – Bank Account System:

Objective:

Develop a **bank account simulation** where the account number and initial deposit are dynamically generated based on the student's roll number. The system should support **deposits, withdrawals, and transaction tracking** while enforcing financial constraints.

Requirements:

- Generate a **unique account number** using the roll number.
- Set the **initial deposit** as the last digit of the roll number multiplied by **1000 PKR**.
- Implement **deposit** and **withdrawal** functions with form inputs.
- Ensure the **deposit** is always the multiple of your roll number only.
- Use **reduce ()** to calculate the total balance from transaction history.
- Display **transaction history** dynamically using **DOM manipulation**.

Constraints:

- **Withdrawals are limited to 80%** of the current balance.
- Use **only vanilla JavaScript, HTML, and CSS.**
- Implement logic using **arrow functions, arrays, map(), reduce(), set, and JavaScript built-in functions**.

Bonus:

- Provide an option to download transaction history as a `.txt` file.

Task 5 – Roll-Number based Discount System:

Objective:

Develop an **e-commerce discount system** where students enter their roll number to get a dynamically generated discount on a product. The roll number determines the discount percentage, and the final price is calculated in real time.

Requirements:

- Extract the **middle two digits** of the roll number to determine the **discount percentage** (e.g., if the roll number is 21F-9445, the discount is 44%).
- Display a list of **products with prices** and allow the student to select one.
- Calculate and show the **discounted price** dynamically using **DOM manipulation**.
- Use **reduce()** to apply multiple discounts if a student selects more than one product.

- Ensure real-time price updates when the student enters their roll number.

Constraints:

- The **maximum discount is 50%**, even if the roll number suggests more.
- Implement logic using **arrays, map(), reduce(), set, and arrow functions**.
- The UI must **update instantly** without a page reload.
- Use **only vanilla JavaScript, HTML, and CSS.**

Bonus:

- Allow students to enter an additional promo code for extra discounts.
- Increase **max discounts to 60%** after 2 purchases.