Set – 8

Q1. Web Frontend (Scenario)
Scenario: You're building a personal productivity app. Create a simple task manager with the following:
• An input field to add a new task.
• An "Add" button that appends the task to a list.
• Each task in the list should have a checkbox to mark it as complete (strikethrough text when checked) and a delete button to remove it.
• Use AI to generate the HTML structure, CSS for styling the completed tasks, and JavaScript for the interactive functionality.
CODE:

Task1.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Simple Task Manager</title>
    <link rel="stylesheet" href="Task1.css">
</head>
<body>
    <h1>Task Manager</h1>

    <div id="task-input-section">
        <input id="task-input" type="text" placeholder="Add a new task..."/>
        <button id="add-btn">Add</button>
    </div>

    <ul id="task-list"></ul>

    <script src="Task1.js"></script>
</body>
</html>
```

CSS:

Task1.css

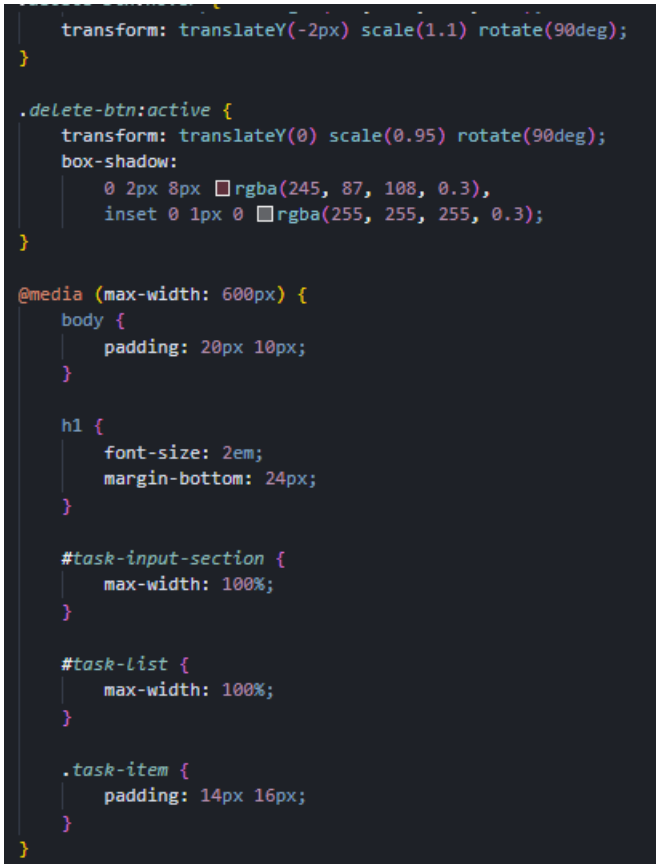
```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 40px 20px;
    min-height: 100vh;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: #222;
    display: flex;
    justify-content: center;
    align-items: flex-start;
    position: relative;
    overflow-x: hidden;
}

body::before {
    content: '';
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background:
        radial-gradient(circle at 20% 50%, rgba(255, 255, 255, 0.1) 0%, transparent 50%),
        radial-gradient(circle at 80% 80%, rgba(255, 255, 255, 0.1) 0%, transparent 50%);
    pointer-events: none;
    z-index: 0;
}

h1 {
    text-align: center;
    margin-bottom: 32px;
    font-size: 2.5em;
    font-weight: 700;
    color: #ffffff;
    text-shadow:
        0 2px 4px rgba(0, 0, 0, 0.3),
```

```
    letter-spacing: 1px;
    position: relative;
    z-index: 1;
}

#task-input-section {
    display: flex;
    justify-content: center;
    margin-bottom: 32px;
    position: relative;
    z-index: 1;
    max-width: 500px;
    margin-left: auto;
    margin-right: auto;
}

#task-input {
    flex-grow: 1;
    padding: 16px 20px;
    font-size: 16px;
    border: none;
    border-radius: 12px 0 0 12px;
    outline: none;
    background: rgba(255, 255, 255, 0.95);
    backdrop-filter: blur(10px);
    box-shadow:
        0 8px 32px rgba(0, 0, 0, 0.1),
        inset 0 1px 0 rgba(255, 255, 255, 0.6);
    transition: all 0.3s ease;
    color: #333;
}

#task-input:focus {
    background: rgba(255, 255, 255, 1);
    box-shadow:
        0 12px 40px rgba(0, 0, 0, 0.15),
        inset 0 1px 0 rgba(255, 255, 255, 0.8),
        0 0 0 3px rgba(102, 126, 234, 0.3);
    transform: translateY(-2px);
}

#add-btn {
```

```css
#add-btn {
    padding: 16px 28px;
    font-size: 16px;
    font-weight: 600;
    color: #fff;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    border: none;
    border-radius: 0 12px 12px 0;
    cursor: pointer;
    transition: all 0.3s ease;
    box-shadow:
        0 8px 24px rgba(102, 126, 234, 0.4),
        inset 0 1px 0 rgba(255, 255, 255, 0.2);
    text-transform: uppercase;
    letter-spacing: 0.5px;
}

#add-btn:hover {
    background: linear-gradient(135deg, #764ba2 0%, #667eea 100%);
    box-shadow:
        0 12px 32px rgba(102, 126, 234, 0.5),
        inset 0 1px 0 rgba(255, 255, 255, 0.3);
    transform: translateY(-2px) scale(1.02);
}

#add-btn:active {
    transform: translateY(0) scale(0.98);
    box-shadow:
        0 4px 16px rgba(102, 126, 234, 0.3),
        inset 0 1px 0 rgba(255, 255, 255, 0.2);
}

#task-list {
    list-style: none;
    padding: 0;
    max-width: 500px;
    margin: 0 auto;
    position: relative;
    z-index: 1;
}

.task-item {
    display: flex;
```

```css
    align-items: center;
    padding: 18px 20px;
    margin-bottom: 16px;
    background: rgba(255, 255, 255, 0.95);
    backdrop-filter: blur(10px);
    border-radius: 16px;
    box-shadow:
        0 8px 32px rgba(0, 0, 0, 0.1),
        inset 0 1px 0 rgba(255, 255, 255, 0.6);
    transition: all 0.3s ease;
    border: 1px solid rgba(255, 255, 255, 0.3);
}

.task-item:hover {
    transform: translateY(-4px) scale(1.02);
    box-shadow:
        0 12px 40px rgba(0, 0, 0, 0.15),
        inset 0 1px 0 rgba(255, 255, 255, 0.8);
    background: rgba(255, 255, 255, 1);
}

.task-item input[type="checkbox"] {
    width: 24px;
    height: 24px;
    cursor: pointer;
    accent-color: #667eea;
    border-radius: 6px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    transition: all 0.3s ease;
}

.task-item input[type="checkbox"]:hover {
    transform: scale(1.1);
    box-shadow: 0 4px 12px rgba(102, 126, 234, 0.3);
}

.task-item label {
    flex-grow: 1;
    margin-left: 16px;
    cursor: pointer;
    user-select: none;
    transition: all 0.3s ease;
```

```css
.task-item label {
    font-size: 16px;
    font-weight: 500;
    color: #333;
}

.completed label {
    text-decoration: line-through;
    color: #999;
    opacity: 0.6;
}

.completed {
    background: rgba(255, 255, 255, 0.7);
    opacity: 0.8;
}

.delete-btn {
    background: linear-gradient(135deg, #f093fb 0%, #f5576c 100%);
    border: none;
    color: #fff;
    font-size: 20px;
    font-weight: bold;
    width: 36px;
    height: 36px;
    margin-left: 12px;
    cursor: pointer;
    border-radius: 50%;
    transition: all 0.3s ease;
    box-shadow:
        0 4px 12px rgba(245, 87, 108, 0.4),
        inset 0 1px 0 rgba(255, 255, 255, 0.3);
    display: flex;
    align-items: center;
    justify-content: center;
    line-height: 1;
}

.delete-btn:hover {
    background: linear-gradient(135deg, #f5576c 0%, #f093fb 100%);
    box-shadow:
        0 6px 20px rgba(245, 87, 108, 0.5),
        inset 0 1px 0 rgba(255, 255, 255, 0.4);
```

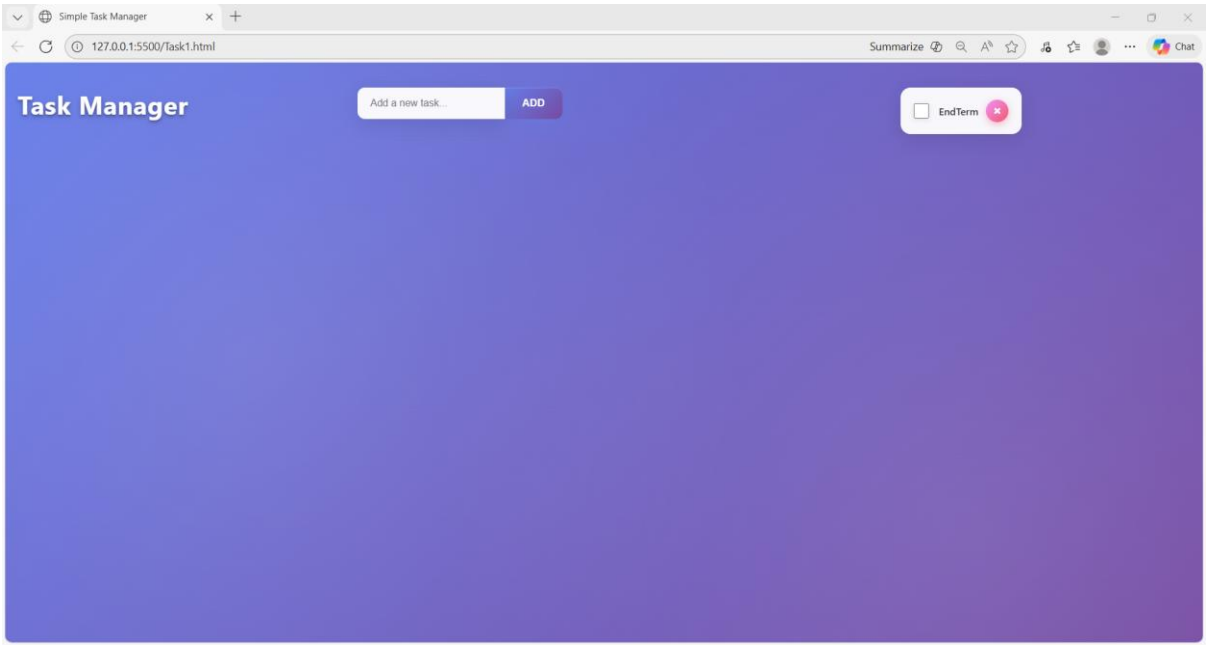


```
    transform: translateY(-2px) scale(1.1) rotate(90deg);
}

.delete-btn:active {
    transform: translateY(0) scale(0.95) rotate(90deg);
    box-shadow:
        0 2px 8px rgba(245, 87, 108, 0.3),
        inset 0 1px 0 rgba(255, 255, 255, 0.3);
}

@media (max-width: 600px) {
    body {
        padding: 20px 10px;
    }

    h1 {
        font-size: 2em;
        margin-bottom: 24px;
    }

    #task-input-section {
        max-width: 100%;
    }

    #task-list {
        max-width: 100%;
    }

    .task-item {
        padding: 14px 16px;
    }
}
```

JAVA SCRIPT:

```
const taskInput = document.getElementById('task-input');
const addBtn = document.getElementById('add-btn');
const taskList = document.getElementById('task-list');

function createTaskElement(taskText) {
    const li = document.createElement('li');
    li.className = 'task-item';

    const checkbox = document.createElement('input');
    checkbox.type = 'checkbox';

    const label = document.createElement('label');
    label.textContent = taskText;

    checkbox.addEventListener('change', function() {
        if (checkbox.checked) {
            li.classList.add('completed');
        } else {
            li.classList.remove('completed');
        }
    });

    const deleteBtn = document.createElement('button');
    deleteBtn.className = 'delete-btn';
    deleteBtn.title = 'Delete task';
    deleteBtn.innerHTML = '&times;';

    deleteBtn.addEventListener('click', function() {
        taskList.removeChild(li);
    });

    label.addEventListener('click', function() { checkbox.click(); });

    li.appendChild(checkbox);
    li.appendChild(label);
    li.appendChild(deleteBtn);

    return li;
}

function addTask() {
    const text = taskInput.value.trim();
    if (text) {
function addTask() {
        const taskEl = createTaskElement(text);
        taskList.appendChild(taskEl);
        taskInput.value = '';
        taskInput.focus();
    }
}

addBtn.addEventListener('click', addTask);
taskInput.addEventListener('keydown', function(e) {
    if (e.key === 'Enter') addTask();
});

taskInput.focus();
```

OUTPUT:




Q2. Data Structures (Scenario)

Scenario: Implement a Queue data structure in Python. The class Queue should have enqueue, dequeue, and front methods. Use AI assistance to write the class. Then, using a few-shot prompt, instruct the AI to modify the queue to have a maximum size (a "bounded queue") and raise an exception if one tries to enqueue when the queue is full.

(Include docs + comments + test cases.)

CODE:

Task2.py > Queue

```python
class QueueFullException(Exception):
    """Raised when enqueue is attempted on a full queue."""
    pass

class QueueEmptyException(Exception):
    """Raised when dequeue or front is attempted on an empty queue."""
    pass

class Queue:
    """
    A simple queue implementation with a maximum (bounded) size.

    Methods:
        - enqueue(item): Add item to the end of the queue. Raises QueueFullException if full.
        - dequeue(): Remove and return the item from the front. Raises QueueEmptyException if empty.
        - front(): Return the item at the front without removing it. Raises QueueEmptyException if empty.
        - is_empty(): Returns True if queue is empty.
        - is_full(): Returns True if queue is full.
    """

    def __init__(self, max_size):
        """
        Initialize the queue with a maximum size.
        :param max_size: int, the maximum number of items the queue can hold.
        """
        self._items = []
        self._max_size = max_size

    def enqueue(self, item):
        """
        Add an item to the back of the queue.

        :param item: The item to add.
```

```
def enqueue(self, item):
    :raises QueueFullException: If the queue is already full.
    """
    if self.is_full():
        raise QueueFullException("Queue is full")
    self._items.append(item)

def dequeue(self):
    """
    Remove and return the item at the front of the queue.

    :raises QueueEmptyException: If the queue is empty.
    :return: The item removed from the queue.
    """
    if self.is_empty():
        raise QueueEmptyException("Queue is empty")
    return self._items.pop(0)

def front(self):
    """
    Return the item at the front of the queue without removing it.

    :raises QueueEmptyException: If the queue is empty.
    :return: The item at the front of the queue.
    """
    if self.is_empty():
        raise QueueEmptyException("Queue is empty")
    return self._items[0]

def is_empty(self):
    """Returns True if the queue is empty, False otherwise."""
    return len(self._items) == 0
```

```
    def is_full(self):
        """Returns True if the queue is full, False otherwise."""
        return len(self._items) >= self._max_size

    def __len__(self):
        """Returns the current number of items in the queue."""
        return len(self._items)

def run_user_defined_tests():
    """
    Interactive harness that lets the user supply their own test cases.

    Workflow:
        1. Prompt for maximum queue size.
        2. Prompt for number of test cases (operations) to run.
        3. For each test case, read an operation such as:
                enqueue <value>
                dequeue
                front
                is_empty
                is_full
                len
        4. Execute the operation, printing the outcome (or any exception encountered)
    """
    print("Queue Test Harness (User-Driven)")
    print("--------------------------------")

    # Step 1: queue capacity
    while True:
        try:
            max_size = int(input("Enter maximum queue size: "))
            if max_size <= 0:
```

```
            raise ValueError("Max size must be positive.")
        break
    except ValueError as exc:
        print(f"Invalid max size. {exc}")

queue = Queue(max_size=max_size)

# Step 2: number of operations
while True:
    try:
        total_cases = int(input("Enter number of test cases (operations): "))
        if total_cases <= 0:
            raise ValueError("Number of test cases must be positive.")
        break
    except ValueError as exc:
        print(f"Invalid count. {exc}")

print(
    "\nSupported commands:\n"
    "  enqueue <value>  -> enqueue a string payload\n"
    "  dequeue          -> dequeue and print the removed value\n"
    "  front            -> show the value at the front without removing it\n"
    "  is_empty         -> prints True/False\n"
    "  is_full          -> prints True/False\n"
    "  len              -> prints the current queue length\n"
)

# Step 3: process user-supplied operations
for idx in range(1, total_cases + 1):
    raw = input(f"Test case {idx}: ").strip()
    if not raw:
        print(f"[{idx}] Skipped (blank input).")
        continue
```

```
            print(f"[{idx}] Skipped (blank input).")
            continue

        parts = raw.split(maxsplit=1)
        command = parts[0].lower()
        payload = parts[1] if len(parts) > 1 else None

        try:
            if command == "enqueue":
                if payload is None:
                    raise ValueError("enqueue requires a value, e.g., 'enqueue apple'.")
                queue.enqueue(payload)
                print(f"[{idx}] enqueue -> '{payload}' inserted.")

            elif command == "dequeue":
                removed = queue.dequeue()
                print(f"[{idx}] dequeue -> '{removed}' removed.")

            elif command == "front":
                front_val = queue.front()
                print(f"[{idx}] front -> '{front_val}'.")

            elif command == "is_empty":
                print(f"[{idx}] is_empty -> {queue.is_empty()}.")

            elif command == "is_full":
                print(f"[{idx}] is_full -> {queue.is_full()}.")

            elif command == "len":
                print(f"[{idx}] len -> {len(queue)}.")

            else:
                raise ValueError(
                    f"Unknown command '{command}'. "
                    "Use enqueue/dequeue/front/is_empty/is_full/len."
                )

        except (QueueFullException, QueueEmptyException, ValueError) as exc:
            print(f"[{idx}] Error: {exc}")

    print("\nAll user-supplied test cases processed.")


if __name__ == "__main__":
    run_user_defined_tests()
```

TEST_CASE:

test_task2.py › ...

```python
"""
Unit tests for the Queue implementation in Task2.py.

Run with:
    python -m unittest test_task2.py
"""

import unittest

from Task2 import (
    Queue,
    QueueFullException,
    QueueEmptyException,
)


class TestQueue(unittest.TestCase):
    """Covers the core queue behaviors and edge cases."""

    def test_enqueue_updates_front(self):
        queue = Queue(max_size=3)
        queue.enqueue("a")
        self.assertEqual(queue.front(), "a")
        self.assertFalse(queue.is_empty())
        self.assertEqual(len(queue), 1)

    def test_fill_queue_to_capacity(self):
        queue = Queue(max_size=3)
        queue.enqueue("a")
        queue.enqueue("b")
        queue.enqueue("c")
        self.assertTrue(queue.is_full())
        self.assertEqual(len(queue), 3)
```

```
class TestQueue(unittest.TestCase):
    def test_enqueue_raises_when_full(self):
        queue = Queue(max_size=1)
        queue.enqueue("a")
        with self.assertRaises(QueueFullException):
            queue.enqueue("b")

    def test_dequeue_fifo(self):
        queue = Queue(max_size=3)
        queue.enqueue("a")
        queue.enqueue("b")
        queue.enqueue("c")
        self.assertEqual(queue.dequeue(), "a")
        self.assertEqual(queue.front(), "b")
        self.assertFalse(queue.is_full())
        self.assertEqual(len(queue), 2)
        self.assertEqual(queue.dequeue(), "b")
        self.assertEqual(queue.dequeue(), "c")
        self.assertTrue(queue.is_empty())

    def test_dequeue_raises_when_empty(self):
        queue = Queue(max_size=1)
        with self.assertRaises(QueueEmptyException):
            queue.dequeue()

    def test_front_raises_when_empty(self):
        queue = Queue(max_size=1)
        with self.assertRaises(QueueEmptyException):
            queue.front()


if __name__ == "__main__":
    unittest.main()
```

OUTPUT:

```
.py
Queue Test Harness (User-Driven)
------------------------------
Enter maximum queue size: 3
Enter number of test cases (operations): 6

Supported commands:
  enqueue <value>  -> enqueue a string payload
  dequeue          -> dequeue and print the removed value
  front            -> show the value at the front without removing it
  is_empty         -> prints True/False
  is_full          -> prints True/False
  len              -> prints the current queue length

Test case 1: enqueue apple
[1] enqueue -> 'apple' inserted.
Test case 2: enqueue banana
[2] enqueue -> 'banana' inserted.
Test case 3: front
[3] front -> 'apple'.
Test case 4: queueue
[4] Error: Unknown command 'queueue'. Use enqueue/dequeue/front/is_empty/is_full/len.
Test case 5: dequeue apple
[5] dequeue -> 'apple' removed.
Test case 6: is_empty
[6] is_empty -> False.

All user-supplied test cases processed.
PS C:\Users\DEEKSHA\OneDrive\Desktop\AIAC\End-term>
```

TEST-CASES OUTPUT :

```
task2.py
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK
```