ASSIGNMENT - 02

CSA4307

INTERNET PROGRAMMING

-192311055
Jacintk Priscilla. J
BE. CSE

Create a simple video game using HTML and Javascript

```
<!Doctype html>
<head>
    <title> click the star game. </title>
    <style>
        body {
        font. family : Arial , sans serif ;
        text. align : center ;
        background. color : #f0f8ff ;
        Overflow : hidden,
    }
    h1 {
        margin . top: 20px ;
    }
    #gameArea{
            Position : relative ;
            width : 100vm ;
            height : 80vh ;
            Margin : auto
        }
        #star {
            Position : absolute ;
            width : 50px ;
            cursor : pointer ;
        }
```

```
#gameover {
    font-size : 1.5cm;
    color : red;
    margin-top : 20px;
    display : none;
    }
</style>
</head>
<body>
<h1> Click the star Game </h1>
<div id = "scoreboard">
    Time left : <span id = "timer"> 30 </span> seconds
    Score : <span id = "score"> 0 </space>
</div>
<div id = "gameArea">
<img src = "https://..."
</div>
<div id = "gameover"> Game over ! final score : <span
    id = "finalscore"> 0 </span> </div>
</script>
const star = document.getElementById('star');
const timerdisplay = document.getElementById('timer');
const scoredisplay = document.getElementById('score');
```

```
const finalscore = document.getElementById('finalscore');
const gameArea = document.getElementById('gameArea');
let score = 0;
let timeLeft = 30;
let gameRunning = true;
function MoveStar(){
    if(!gameRunning) return;
    const maxX = gameArea.clientwidth - star.clientwidth;
    const maxY = gameArea.clientwidth - star.clientheight;
    const randomX = math.random() * maxX;
    const randomY = math.random() * maxY;
    star.style.left = `${random}px`;
    star.style.top = `${randomy}px`;
}
function updateTimer(){
    timeLeft--;
    timeDisplay.textContent = timeleft;
    if(timeleft <= 0){
    clearInterval(timeInterval);
    gameRunning = false;
    star.style.display = "none";
```

```
finalscore.textContent = score;
gameoverMessage.style.display = "block";
}
}
const timeInterval = setInterval(updateTimer, 1000);
moveStar();
setInterval(() => {
    if (gameRunning) moveStar();
}, 700);
</script>
</body>
</html>
```

Creating an Application

```
<!Doctype html>
<head>
<title> TO. Do list </title>
<link rel = "stylesheet" href = "style.css">
</head>
<body>
<div class = "container">
<h1> TO-do list </h1>
```

2) Displaying Tasks

```
function renderTask(){
  const taskList = document.getElementById
                   ('taskList');
taskList.innerHTML = '';
tasks.forEach((task, index) => {
  const li = document.createElement('li');
  li.className = task.completed ? 'complet : complet
  li.innerHTML = '
  <strong>${task.title}</strong><br>
  <span>${task.description}</span>
  <div class="task actions">
  <button onclick="editTask(${index})">Edition
                                  </button>
  <button onclick="deleteTask(${index})">delete
                                  </button>
```

Managing Tasks

1) add new Task

```
document.getElementById("taskform").addEventListner
        ('submit', e => { e.preventDefault();
const title = document.getElementById('title').
              value.trim();
```

```
<input type="text" id="description" placeholder="Task
                                        description">
<button type="submit">Add Task</button>
</form>
  <ul id="tasklist"></ul>
</div>
<script src="script.js"></script>
</body>
</html>
```

Fetching Tasks

```
window.addEventListner("DomContent.loaded",()=>{
  fetch('task.json')
  .then(response=>response.json())
  .then(data=>{
      tasks=data;
      renderTasks();
  });
});
```

```
const description = document.getElementById('description').
                                    value.trim();
if (title && description)
  {
    tasks.push({ title, description, completed: false});
    renderTasks();
  }
});
```

99) Edit Task

```
function editTask(index){
  const newTitle = prompt("Edit Title", tasks[index].title);
  const newDesc = prompt("Edit description:", tasks[index];
  if(newTitle != null && newDesc != Null)
{
  tasks[index].title = newTitle.trim();
  tasks[index].description = newDesc.trim();
  renderTasks();
}
}
```

99) Remove Task

```
function deleteTask(index)
{
  tasks.splice(index, 1);
  renderTasks();
}
```

3r) Toggle complete / Incomple.

```
function togglestatus (Index)
{
tasks [Index] . completed = ! tasks [index] completed;
renderTask();
}
```

4) User Interface.

(HTML + CSS snippet)

HTML

```
<form id = "taskform">
<input type = "text" id = "title" placeholder = "task Title">
<input type = "text" id = "description" placeholder = "Task
                                                    description">
<button type = "submit"> Add Task </button>
</form>
<ul id = "taskList"></ul>
```

CSS

```
form
{
  display : flex;
  gap : 10px;
  margin - bottom : 20px;
```

```
}
button {
    Padding : 8px;
    cursor : pointer;
}
```

5. functionality

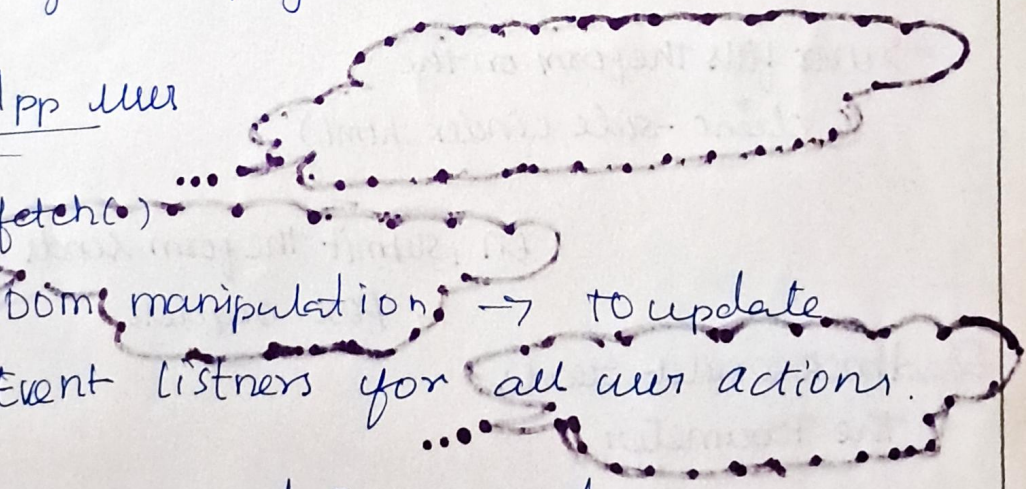
Requirement : All interactions must work with Javascript without reloading the page.

i) The App user

=> fetch()

=> DOM manipulation -> to update

=> Event listners for all user actions.



ii) No page reload is needed

3) Client Server Communication

9) The form uses post to submit to the feedbackServlet

99) The server handles validation & return message

* A thank-you on success
* An Error message if fields are missing

Explanation of Flow.