# WEB-TECHNOLOGY PROJECT DOCUMENTARY

Presented By : Beksultan Moldakhmet

# PROJECT OVERVIEW

- **Project Name: Memory Game**
- **Description: A memory game where players need to find pairs of matching cards.**
- **Technologies Used: HTML, CSS, JavaScript**

# PROJECT GOALS

create a memory game where users can select a difficulty level.

Implemented with HTML, CSS, and JavaScript:

- Dynamic card creation.
- Timer functionality.
- Displaying win/loss messages.

# TECHNOLOGY USED

## HTML

Structure of the page, interface elements.

## CSS

Styling for cards, buttons, background, etc.

## JAVASCRIPT

Game logic, event handling, timer management, card matching validation.

# HTML - PAGE STRUCTURE

## EXPLANATION:

- : The main container for the entire page.
- <h1>: Title of the game.
- : Container for control buttons.
- Difficulty buttons: Elements for selecting the game difficulty (Easy, Medium, Hard).
- : Where the cards will be displayed.
- : Displays the win or loss message.
- : Displays the game timer.

```
<div class="container">
    <h1>Memory Game</h1> <!-- Заголовок игры -->

    <div class="controls">
        <!-- Кнопка для начала игры -->
        <button id="startBtn">Start Game</button>

        <div id="difficulty">
            <!-- Кнопки для выбора уровня сложности -->
            <button id="easyBtn">Easy</button>
            <button id="mediumBtn">Medium</button>
            <button id="hardBtn">Hard</button>
        </div>
    </div>

    <!-- Игровая доска, где будут отображаться карточки -->
    <div id="gameBoard" class="game-board"></div>

    <!-- Место для вывода сообщений (например, когда игра закончена) -->
    <div id="message"></div>

    <!-- Таймер, который отслеживает время игры -->
    <div id="timer">Time: 0</div>
</div>
```

# CSS - PAGE STYLING

## EXPLANATION:

- body {}: Styles for the page background and layout.
- Gradient background and Flexbox for centering the content.
- button {}: Button styling.
- Button background color, font size, rounded corners, and hover effect.
- .card {}: Card styling.
- Card size, border radius, and center alignment of card content.

```css
body {
    font-family: Arial, sans-serif;
    background: linear-gradient(to right, #6a11cb, #2575fc); /* Градиентный фон */
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    color: #fff;
}

/* Стиль для кнопок */
button {
    background-color: #4CAF50;
    color: white;
    border: none;
    padding: 15px 32px;
    text-align: center;
    font-size: 16px;
    cursor: pointer;
    border-radius: 5px;
    margin: 5px;
}
```

```css
.card {
    width: 100px;
    height: 100px;
    border-radius: 10px;
    background-color: #fff;
    display: flex;
    justify-content: center;
    align-items: center;
    position: relative;
    overflow: hidden;
    cursor: pointer;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    transition: transform 0.3s, box-shadow 0.3s;
}
```

# JAVASCRIPT - STARTING THE GAME

## EXPLANATION:

- startButton.addEventListener('click', () => { ... }): Adds a click event listener to the "Start Game" button.
- Inside the event handler, it checks if the game has started (gameStarted):
  - If not, the game starts and the board is created.
  - If the game has already started, it resets the game.

```
startButton.addEventListener('click', () => {
    if (!gameStarted) {
        gameStarted = true;
        matchedCards = [];
        flippedCards = [];
        message.textContent = '';
        message.classList.remove('lost');
        startButton.textContent = 'Restart Game';  // Изменяю текст кнопки на "Restart Game"
        difficultyDiv.style.display = 'none';  // Скрываюкнопки сложности
        clearInterval(interval);
        interval = setInterval(updateTimer, 1000); // Запускаю таймер
        createBoard();
    } else {
        // Перезапуск игры
        resetGame();
    }
});
```

# JAVASCRIPT - CREATING THE GAME BOARD

## EXPLANATION:

- gameBoard.innerHTML = '';: Clears the game board before adding new cards.
- shuffleCards(cardImages): Shuffles the card images to randomize their placement on the board.
- card.addEventListener('click', flipCard);: Adds a click event listener to flip the card when clicked.
- gameBoard.appendChild(card);: Adds the card to the board.

```
function createBoard() {
    gameBoard.innerHTML = '';  // Очищаю поле перед добавле
    cards = [];
    matchedCards = [];
    flippedCards = [];
    const shuffledCards = shuffleCards(cardImages);

    shuffledCards.forEach((value, index) => {
        const card = document.createElement('div');
        card.classList.add('card');
        card.setAttribute('data-index', index);
        card.dataset.value = value;
        card.addEventListener('click', flipCard);

        gameBoard.appendChild(card);
        cards.push(card);
    });
}
```

# JAVASCRIPT - CHECKING FOR MATCHES

## EXPLANATION:

- flippedCards[0].dataset.value === flippedCards[1].dataset.value: Checks if the flipped cards match.
- If they match, they are added to the matchedCards array and marked as matched.
- If not, the cards are flipped back after a short delay (1 second).

```
function checkMatch() {
    if (flippedCards[0].dataset.value === flippedCards[1].dataset.value) {
        matchedCards.push(flippedCards[0], flippedCards[1]);
        flippedCards.forEach(card => card.classList.add('matched'));
        flippedCards = [];
        if (matchedCards.length === cards.length) {
            clearInterval(interval);
            message.textContent = `You win! Time: ${timer} seconds`;
        }
    } else {
        setTimeout(() => {
            flippedCards.forEach(card => {
                card.classList.remove('flipped');
                const img = card.querySelector('img');
                if (img) img.remove();  // Убираю изображение
            });
            flippedCards = [];
        }, 1000);
    }
}
```

# JAVASCRIPT - TIMER

## EXPLANATION:

- timer--;: Decreases the timer by one every second.
- clearInterval(interval);: Stops the timer when time runs out.
- message.textContent = 'Time is up! You lost!';: Displays a "Time is up!" message when the timer reaches zero.

```
function updateTimer() {
    timer--;
    timerDisplay.textContent = `Time: ${timer}`;
    if (timer <= 0) {
        clearInterval(interval);
        message.textContent = 'Time is up! You lost!';
        message.classList.add('lost');
    }
}
```

# CONCLUSION

- **The project successfully implements a memory game using HTML, CSS, and JavaScript.**
- **The game allows players to select a difficulty level, with dynamic card creation and a timer.**