INTRODUCTION

NAME : TASHI VERMA

•University Roll No: 2300290120260

•Branch: Computer Science

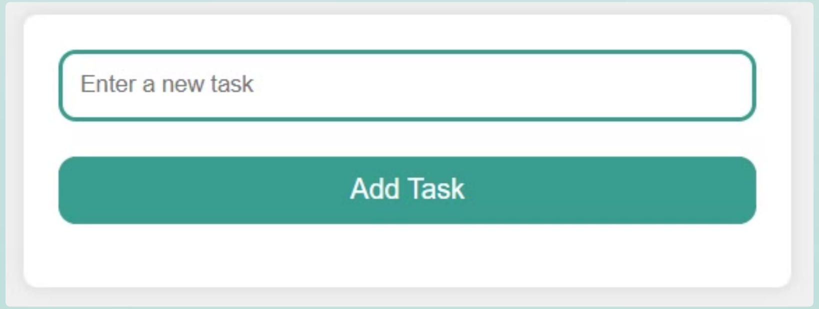
•Year: 2

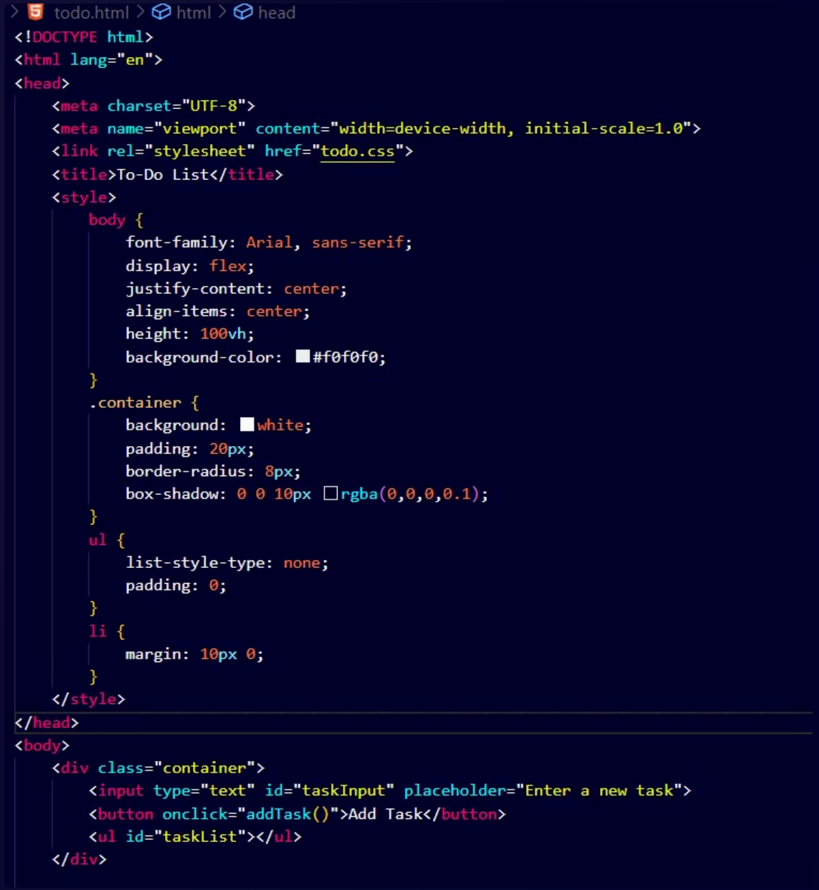
•Section: D

To-Do List Web Application

This presentation outlines the development of a To-Do List web application using HTML, CSS, and JavaScript. The application provides users with a user-friendly interface to manage their tasks effectively.

by Tashi Verma





```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="todo.css">
    <title>To-Do List</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #f0f0f0;
        }
        .container {
            background: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        ul {
            list-style-type: none;
            padding: 0;
        }
        li {
            margin: 10px 0;
        }
    </style>
</head>
<body>
    <div class="container">
        <input type="text" id="taskInput" placeholder="Enter a new task">
        <button onclick="addTask()">Add Task</button>
        <ul id="taskList"></ul>
    </div>
```

Introduction to the Project

1 Overview

A To-Do List application is a valuable tool for individuals and teams seeking to enhance their productivity and time management skills.

2 Purpose

The application allows users to create, organize, and track their tasks, promoting efficient task completion and reduced stress.

3 Target Audience

The target audience includes students, professionals, and anyone who wants to streamline their to-do lists and stay organized.

Key Features of the Application

Task Creation

Users can easily add new tasks with descriptive titles, due dates, and optional priority levels.

Task Editing

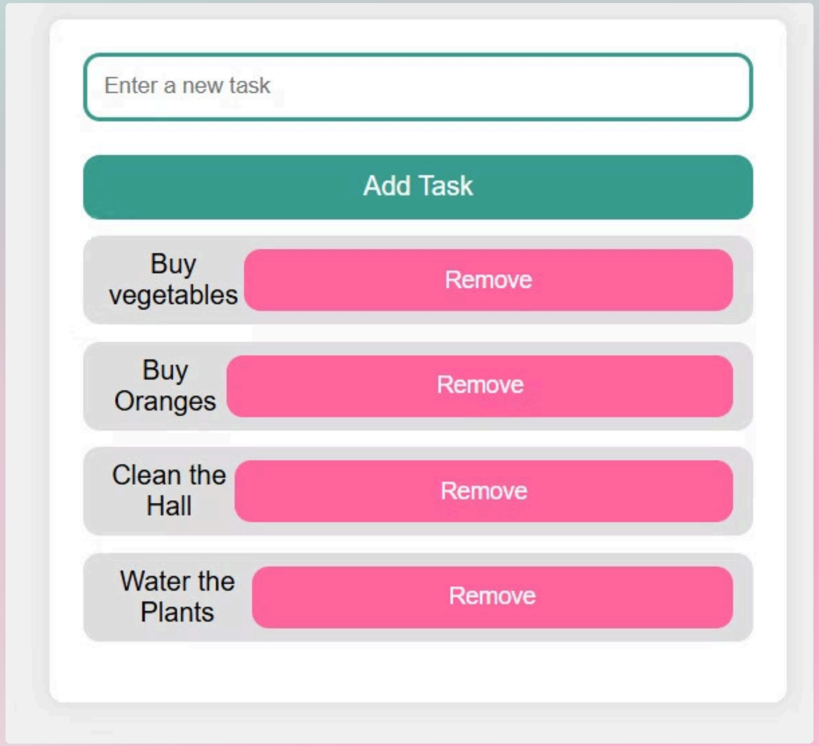
Existing tasks can be modified, including updates to titles, deadlines, and completion status.

Task Deletion

Users can remove completed or irrelevant tasks to maintain a clean and focused list.

Task Completion Marking

Users can mark tasks as complete, providing a visual indication of progress and satisfaction.



HTML Structure and Markup

Structure

The HTML structure defines the overall layout and organization of the web page. It includes a header, main content area, and footer sections.

Markup

Specific HTML elements such as lists (UL and LI), input fields (INPUT), and buttons (BUTTON) are used to create the visual components of the to-do list.

Organization

Tasks are displayed in a list format, allowing users to view, add, and manage their to-dos in a structured manner.

Styling with CSS

1 **Visual Appeal**

CSS is used to enhance the appearance and user experience of the to-do list application.

2 **Layout and Spacing**

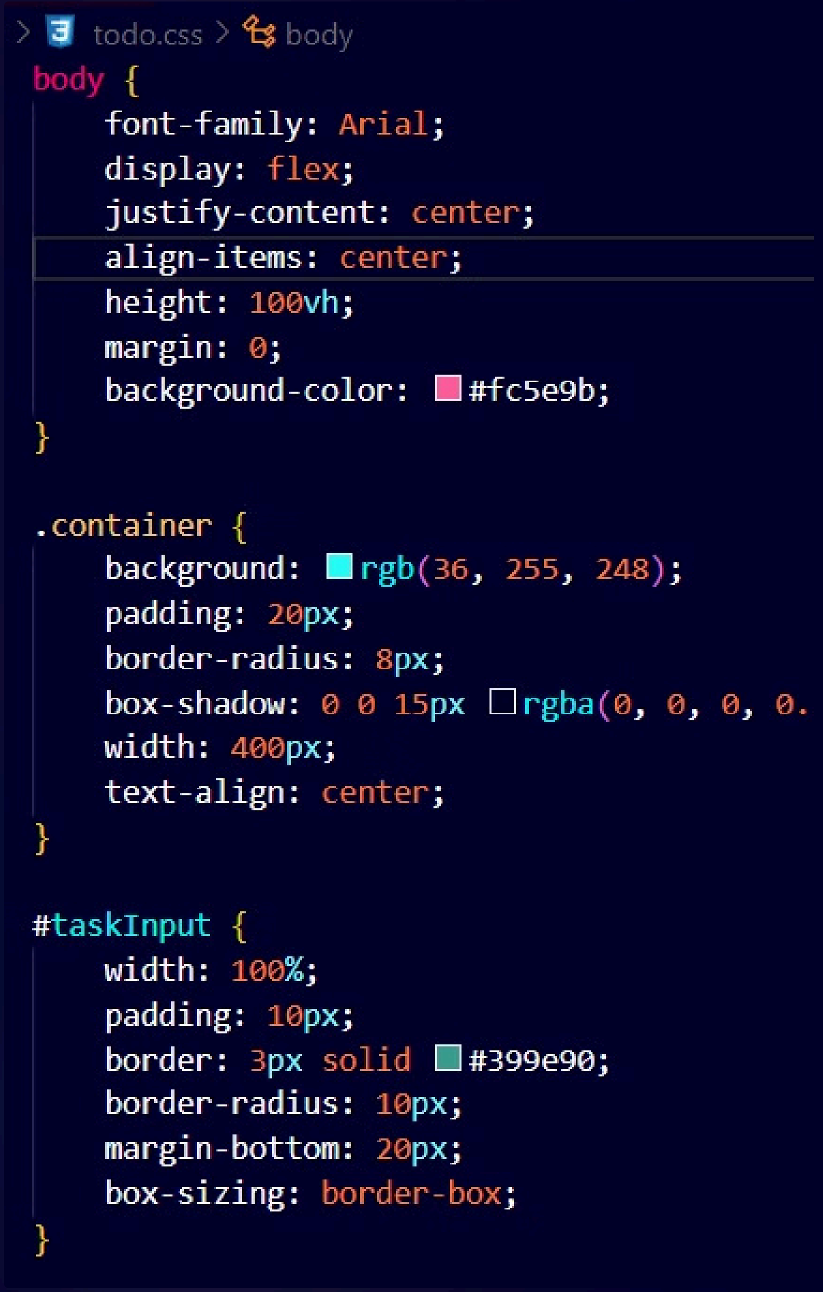
CSS controls the arrangement of elements, ensuring a visually appealing and user-friendly layout.

3 **Color Scheme**

CSS defines the color palette, creating a visually cohesive and consistent design.

4 **Typography**

CSS sets the fonts, font sizes, and styles for text elements, ensuring readability and visual appeal.

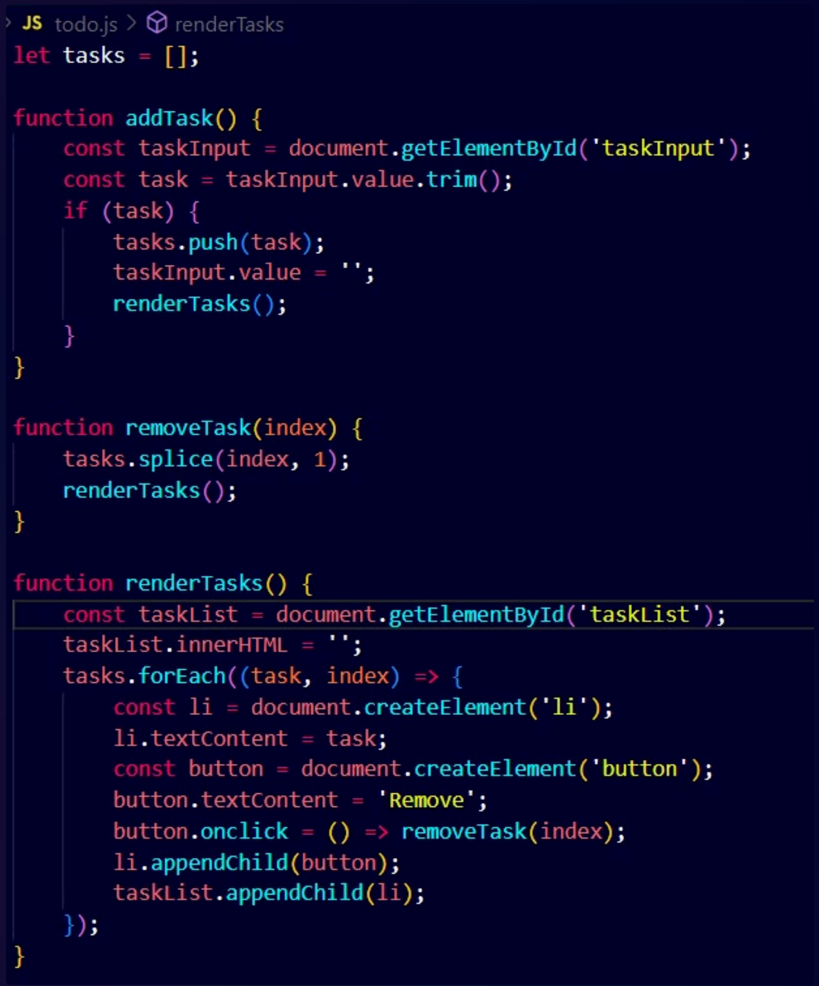


```css
body {
    font-family: Arial;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    background-color: #fc5e9b;
}

.container {
    background: rgb(36, 255, 248);
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.
    width: 400px;
    text-align: center;
}

#taskInput {
    width: 100%;
    padding: 10px;
    border: 3px solid #399e90;
    border-radius: 10px;
    margin-bottom: 20px;
    box-sizing: border-box;
}
```

Made with Gamma

Implementing Functionality with JavaScript

Functionality	Description
Task Addition	JavaScript code handles the process of capturing user input and adding new tasks to the list.
Task Editing	JavaScript enables users to modify existing task details, updating titles, deadlines, and completion status.
Task Deletion	JavaScript implements functionality to remove tasks from the list based on user interaction.
Task Completion	JavaScript allows users to mark tasks as complete, dynamically updating their status in the list.



```javascript
let tasks = [];

function addTask() {
    const taskInput = document.getElementById('taskInput');
    const task = taskInput.value.trim();
    if (task) {
        tasks.push(task);
        taskInput.value = '';
        renderTasks();
    }
}

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

function renderTasks() {
    const taskList = document.getElementById('taskList');
    taskList.innerHTML = '';
    tasks.forEach((task, index) => {
        const li = document.createElement('li');
        li.textContent = task;
        const button = document.createElement('button');
        button.textContent = 'Remove';
        button.onclick = () => removeTask(index);
        li.appendChild(button);
        taskList.appendChild(li);
    });
}
```

Handling User Interactions

Click Events

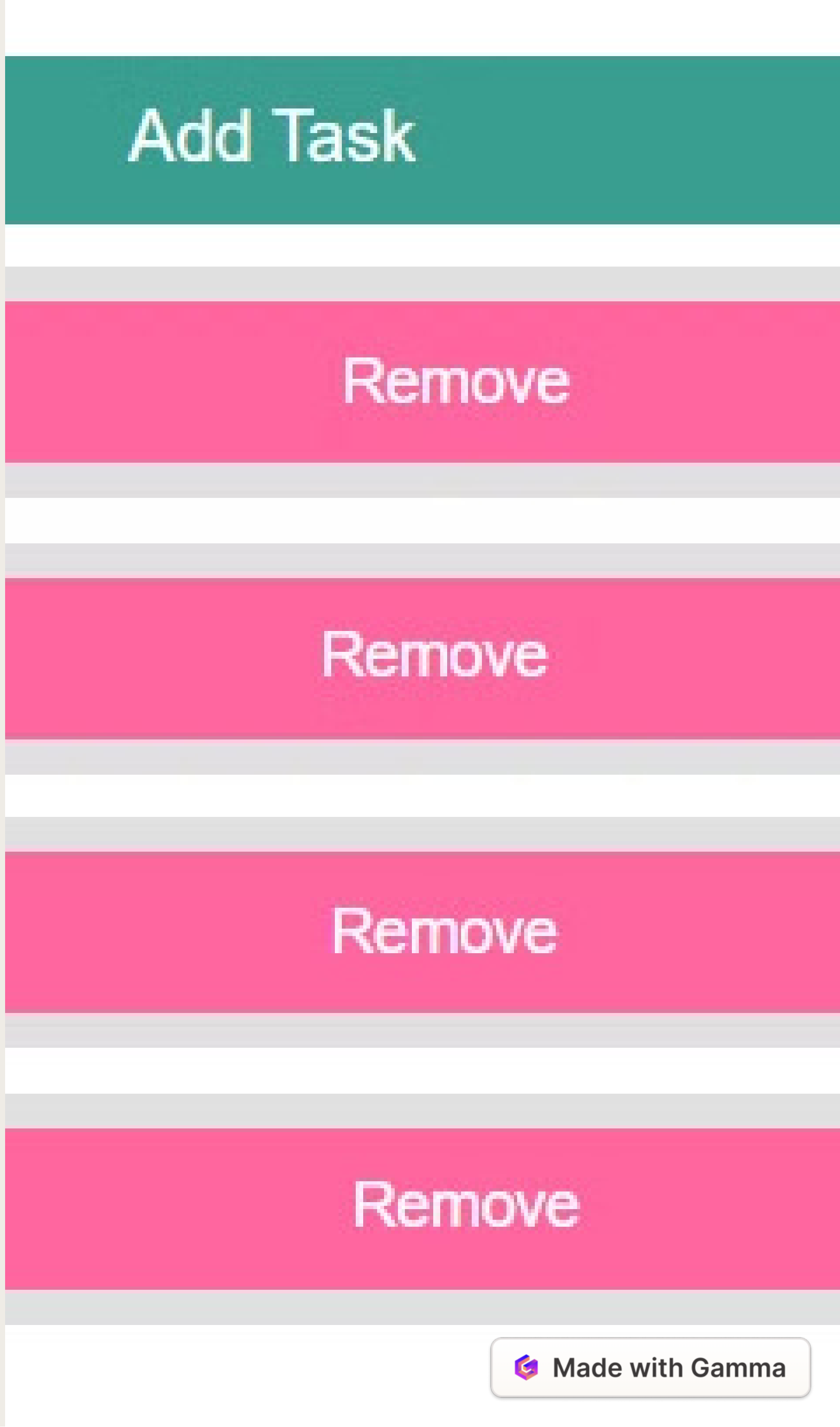
JavaScript listens for click events on buttons and links to trigger specific actions, such as adding, editing, or deleting tasks.

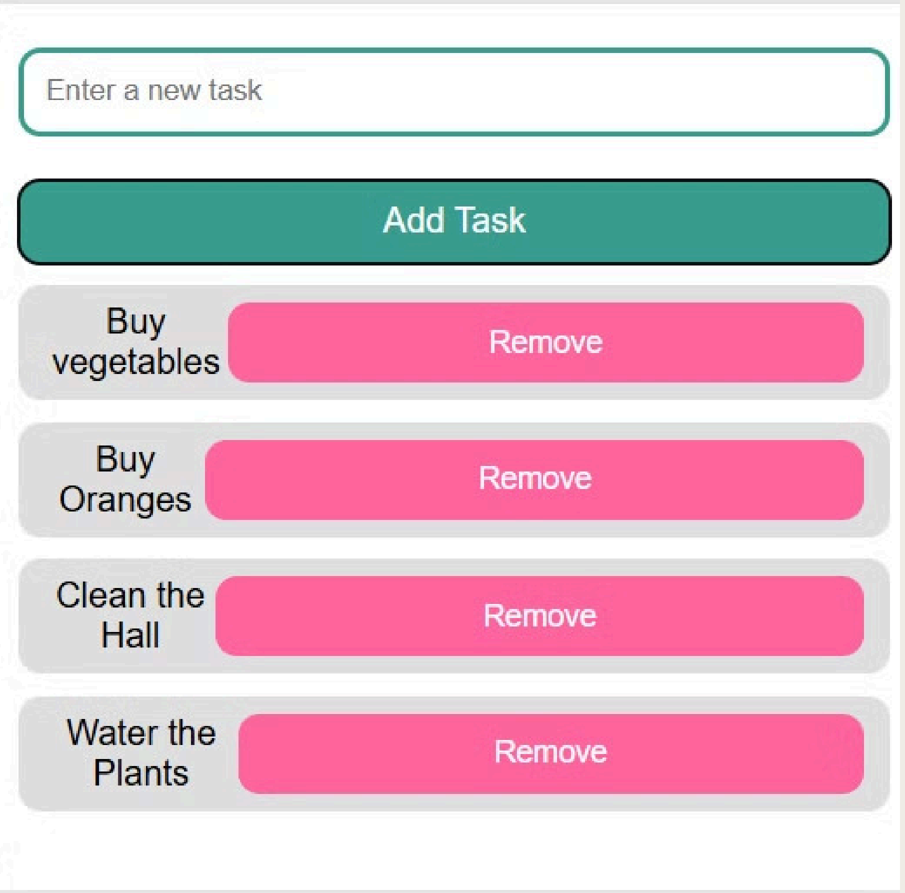
Input Events

JavaScript handles events triggered by user input in text fields, capturing task titles and descriptions.

Keyboard Events

JavaScript responds to keyboard events, such as pressing the Enter key, to execute actions like adding tasks.





Conclusion and Next Steps

1

Future Enhancements

Future enhancements might include features like task prioritization, reminders, and integration with external calendars.

2

Testing and Deployment

The application will undergo thorough testing to ensure functionality and user experience before deployment.

3

User Feedback

User feedback will be gathered to identify areas for improvement and guide further development.