PROJECT : TODO APP

1.INTRODUCTION:

To Do List App is a kind of app that generally used to maintain our day-to-day tasks or list everything that we have to do, with the most important tasks at the top of the list, and the least important tasks at the bottom. It is helpful in planning our daily schedules. We can add more tasks at any time and delete a task that is completed.

2.ABSTRACT:

This abstract provides an overview of the process involved in building a Todo app—a versatile and essential application for managing tasks and increasing productivity. The Todo app is designed to help users organize their daily activities, set reminders, prioritize tasks, and track their progress effectively. Users can create, edit, and delete tasks, set due dates, and categorize tasks into various lists or tags. the app offers a user-friendly interface to view tasks in different formats, such as a list, calendar, or kanban board. The implementation of the Todo app involves utilizing modern web and mobile technologies. The front-end development will employ HTML, CSS, and JavaScript, with popular frameworks like React or Vue.js to enhance user interaction and responsiveness.

3.purpose of the project:

ToDo List App is a kind of app that generally used to maintain our day-to-day tasks or list everything that we have to do, with the most important tasks at the top of the list, and the least important tasks at the bottom. It is helpful in planning our daily schedules. We can add more tasks at any time and delete a task that is completed.

4.Features:

a.Create (add) a new task or adding a new ToDo in the ToDo List App.

b.See all the tasks or View all the ToDos that were added to the app.

c.Delete any ToDo from the list of ToDos.

5.PROCESS

5.1:SET UP THE PROJECT STRUCTURE:

In this phase of developing the todo list app, you will establish the fundamental framework and organization for the project. Create a directory structure that includes essential folders like "src" for source code, "public" for static files, and potentially others for assets or configuration. Set up necessary build tools and dependencies using tools like

Node.js and npm (Node Package Manager). Organize the project to ensure modularity, maintainability, and scalability for future development.

5.2:DESIGN AND IMPLEMENT THE USER INTERFACE:

In this stage, you'll focus on the visual representation and interaction of the todo list app. Design a user-friendly interface that allows users to easily add, edit, and delete tasks. Decide on a consistent and appealing layout, typography, and color scheme. Implement the UI using modern web technologies like HTML, CSS, and JavaScript (and possibly frameworks like React, Angular, or Vue.js). Aim for a responsive design to ensure the app works well on various devices and screen sizes.

5.3:IMPLEMENT TASK MANAGEMENT FUNCTIONALITY:

This step involves adding the core functionality of the todo list app. Develop features that enable users to create new tasks, mark tasks as completed, and update existing tasks. Implement task prioritization, due dates, and task descriptions if needed. Set up data structures to store and manage tasks efficiently. Integrate user interactions with the backend to handle task creation, retrieval, and updates. Ensure that users can perform these operations smoothly and without errors.

5.4:IMPLEMENT TASK FILTERING:

Adding task filtering functionality enhances the user experience by allowing users to view tasks based on specific criteria. Implement filters like "All Tasks," "Completed Tasks," "Active Tasks," and potentially others based on due date, priority, or categories. Make sure the filtering process is intuitive and responsive, providing real-time updates as users select different filters. Efficiently handle the filtering logic on the client-side or interact with the server to retrieve filtered tasks.

6.CODES:

6.1:HTML code:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Todo List App</title>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <link rel="stylesheet" href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">
  </head>
  <body>
    <div class="wrapper">
      <div class="task-input">
        <img src="bars-icon.svg/hi.png" height="45" width="45">
        <input type="text" placeholder="Add a new task">
      </div>
      <div class="controls">
        <div class="filters">
          <span class="active" id="all">All</span>
          <span id="pending">Pending</span>
          <span id="completed">Completed</span>
        </div>
        <button class="clear-btn">Clear All</button>
      </div>
      <ul class="task-box"></ul>
    </div>

    <script src="script.js"></script>

  </body>
</html>
```

6.2:CSS code:

```
/* Import Google Font - Poppins */
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap');
*{
  margin: 0;
```

```
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
body{
  width: 100%;
  height: 100vh;
  overflow: hidden;
  background: linear-gradient(135deg, #4AB1FF, #2D5CFE);
}
::selection{
  color: #fff;
  background: #3C87FF;
}
.wrapper{
  max-width: 405px;
  padding: 28px 0 30px;
  margin: 137px auto;
  background: #fff;
  border-radius: 7px;
  box-shadow: 0 10px 30px rgba(0,0,0,0.1);
}
.task-input{
  height: 52px;
  padding: 0 25px;
  position: relative;
}
.task-input img{
  top: 50%;
  position: absolute;
```

```
  transform: translate(17px, -50%);
}
.task-input input{
  height: 100%;
  width: 100%;
  outline: none;
  font-size: 18px;
  border-radius: 5px;
  padding: 0 20px 0 53px;
  border: 1px solid #999;
}
.task-input input:focus,
.task-input input.active{
  padding-left: 52px;
  border: 2px solid #3C87FF;
}
.task-input input::placeholder{
  color: #bfbfbf;
}
.controls, li{
  display: flex;
  align-items: center;
  justify-content: space-between;
}
.controls{
  padding: 18px 25px;
  border-bottom: 1px solid #ccc;
}
.filters span{
  margin: 0 8px;
```

```
  font-size: 17px;
  color: #444444;
  cursor: pointer;
}
.filters span:first-child{
  margin-left: 0;
}
.filters span.active{
  color: #3C87FF;
}
.controls .clear-btn{
  border: none;
  opacity: 0.6;
  outline: none;
  color: #fff;
  cursor: pointer;
  font-size: 13px;
  padding: 7px 13px;
  border-radius: 4px;
  letter-spacing: 0.3px;
  pointer-events: none;
  transition: transform 0.25s ease;
  background: linear-gradient(135deg, #1798fb 0%, #2D5CFE 100%);
}
.clear-btn.active{
  opacity: 0.9;
  pointer-events: auto;
}
.clear-btn:active{
  transform: scale(0.93);
```

```
}
.task-box{
  margin-top: 20px;
  margin-right: 5px;
  padding: 0 20px 10px 25px;
}
.task-box.overflow{
  overflow-y: auto;
  max-height: 300px;
}
.task-box::-webkit-scrollbar{
  width: 5px;
}
.task-box::-webkit-scrollbar-track{
  background: #f1f1f1;
  border-radius: 25px;
}
.task-box::-webkit-scrollbar-thumb{
  background: #e6e6e6;
  border-radius: 25px;
}
.task-box .task{
  list-style: none;
  font-size: 17px;
  margin-bottom: 18px;
  padding-bottom: 16px;
  align-items: flex-start;
  border-bottom: 1px solid #ccc;
}
.task-box .task:last-child{
```

```
  margin-bottom: 0;
  border-bottom: 0;
  padding-bottom: 0;
}
.task-box .task label{
  display: flex;
  align-items: flex-start;
}
.task label input{
  margin-top: 7px;
  accent-color: #3C87FF;
}
.task label p{
  user-select: none;
  margin-left: 12px;
  word-wrap: break-word;
}
.task label p.checked{
  text-decoration: line-through;
}
.task-box .settings{
  position: relative;
}
.settings :where(i, li){
  cursor: pointer;
}
.settings .task-menu{
  z-index: 10;
  right: -5px;
  bottom: -65px;
```

```
  padding: 5px 0;
  background: #fff;
  position: absolute;
  border-radius: 4px;
  transform: scale(0);
  transform-origin: top right;
  box-shadow: 0 0 6px rgba(0,0,0,0.15);
  transition: transform 0.2s ease;
}
.task-box .task:last-child .task-menu{
  bottom: 0;
  transform-origin: bottom right;
}
.task-box .task:first-child .task-menu{
  bottom: -65px;
  transform-origin: top right;
}
.task-menu.show{
  transform: scale(1);
}
.task-menu li{
  height: 25px;
  font-size: 16px;
  margin-bottom: 2px;
  padding: 17px 15px;
  cursor: pointer;
  justify-content: flex-start;
}
.task-menu li:last-child{
  margin-bottom: 0;
```

```
}
.settings li:hover{
  background: #f5f5f5;
}
.settings li i{
  padding-right: 8px;
}

@media (max-width: 400px) {
  body{
    padding: 0 10px;
  }
  .wrapper {
    padding: 20px 0;
  }
  .filters span{
    margin: 0 5px;
  }
  .task-input{
    padding: 0 20px;
  }
  .controls{
    padding: 18px 20px;
  }
  .task-box{
    margin-top: 20px;
    margin-right: 5px;
    padding: 0 15px 10px 20px;
  }
  .task label input{
```

```
    margin-top: 4px;
  }
}
```

6.3:JS code:

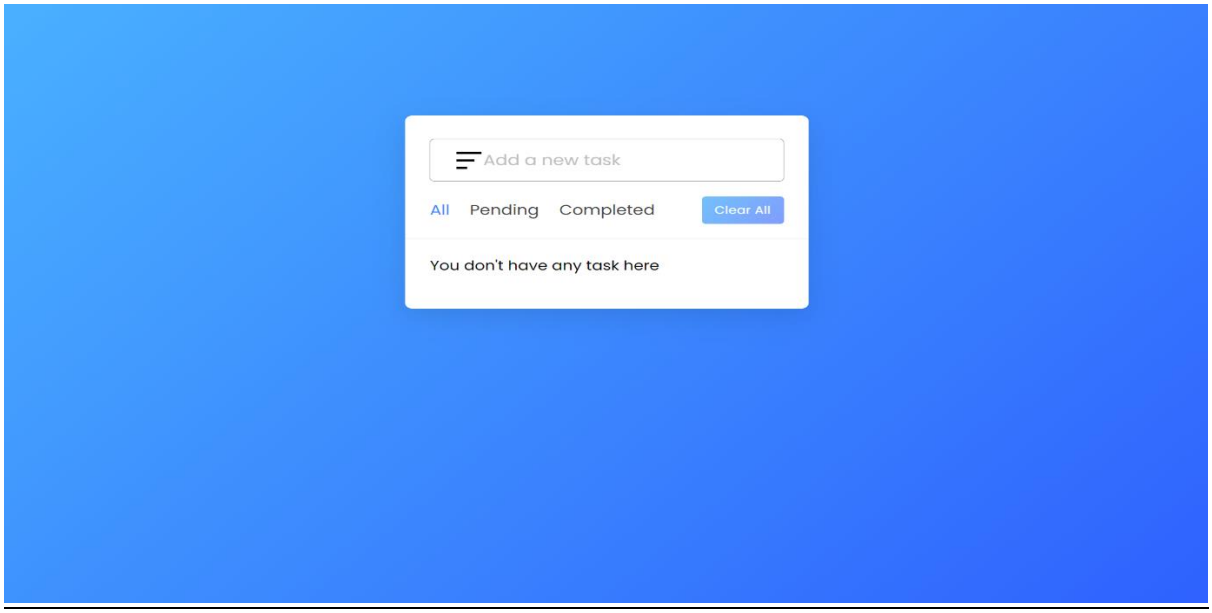
```
const taskInput = document.querySelector(".task-input input"),
filters = document.querySelectorAll(".filters span"),
clearAll = document.querySelector(".clear-btn"),
taskBox = document.querySelector(".task-box");

let editId,
isEditTask = false,
todos = JSON.parse(localStorage.getItem("todo-list"));

filters.forEach(btn => {
    btn.addEventListener("click", () => {
        document.querySelector("span.active").classList.remove("active");
        btn.classList.add("active");
        showTodo(btn.id);
    });
});

function showTodo(filter) {
    let liTag = "";
    if(todos) {
        todos.forEach((todo, id) => {
            let completed = todo.status == "completed" ? "checked" : "";
            if(filter == todo.status || filter == "all") {
                liTag += `<li class="task">
                        <label for="${id}">
```

```
                    <input onclick="updateStatus(this)" type="checkbox" id="${id}"
${completed}>
                    <p class="${completed}">${todo.name}</p>
                </label>
                <div class="settings">
                    <i onclick="showMenu(this)" class="uil uil-ellipsis-h"></i>
                    <ul class="task-menu">
                        <li onclick='editTask(${id}, "${todo.name}")'><i class="uil uil-
pen"></i>Edit</li>
                        <li onclick='deleteTask(${id}, "${filter}")'><i class="uil uil-
trash"></i>Delete</li>
                    </ul>
                </div>
            </li>`;
        }
    });
  }
  taskBox.innerHTML = liTag || `<span>You don't have any task here</span>`;
  let checkTask = taskBox.querySelectorAll(".task");
  !checkTask.length ? clearAll.classList.remove("active") : clearAll.classList.add("active");
  taskBox.offsetHeight >= 300 ? taskBox.classList.add("overflow") :
taskBox.classList.remove("overflow");
}
showTodo("all");

function showMenu(selectedTask) {
  let menuDiv = selectedTask.parentElement.lastElementChild;
  menuDiv.classList.add("show");
  document.addEventListener("click", e => {
    if(e.target.tagName != "I" || e.target != selectedTask) {
      menuDiv.classList.remove("show");
    }
```

```
    });
}

function updateStatus(selectedTask) {
    let taskName = selectedTask.parentElement.lastElementChild;
    if(selectedTask.checked) {
        taskName.classList.add("checked");
        todos[selectedTask.id].status = "completed";
    } else {
        taskName.classList.remove("checked");
        todos[selectedTask.id].status = "pending";
    }
    localStorage.setItem("todo-list", JSON.stringify(todos))
}

function editTask(taskId, textName) {
    editId = taskId;
    isEditTask = true;
    taskInput.value = textName;
    taskInput.focus();
    taskInput.classList.add("active");
}

function deleteTask(deleteId, filter) {
    isEditTask = false;
    todos.splice(deleteId, 1);
    localStorage.setItem("todo-list", JSON.stringify(todos));
    showTodo(filter);
}
```

```
clearAll.addEventListener("click", () => {
    isEditTask = false;
    todos.splice(0, todos.length);
    localStorage.setItem("todo-list", JSON.stringify(todos));
    showTodo()
});

taskInput.addEventListener("keyup", e => {
    let userTask = taskInput.value.trim();
    if(e.key == "Enter" && userTask) {
        if(!isEditTask) {
            todos = !todos ? [] : todos;
            let taskInfo = {name: userTask, status: "pending"};
            todos.push(taskInfo);
        } else {
            isEditTask = false;
            todos[editId].name = userTask;
        }
        taskInput.value = "";
        localStorage.setItem("todo-list", JSON.stringify(todos));
        showTodo(document.querySelector("span.active").id);
    }
});
```

Output:



8.Advantages of a To-Do App:

a.Task Organization: To-do apps allow users to organize their tasks efficiently, creating lists, categories, or tags to prioritize and categorize tasks according to their importance and deadlines.

b.Accessibility: Most to-do apps are available on multiple platforms, including smartphones, tablets, and desktop computers, making it easy for users to access their tasks from anywhere with an internet connection.

c.Reminders and Notifications: To-do apps often come with built-in reminders and notifications, helping users stay on top of their tasks and deadlines. These reminders can be set up to notify users at specific times or when they are in a particular location.

d.Collaboration: Many to-do apps offer collaboration features, allowing users to share task lists with others, delegate tasks, and track progress together. This is particularly useful for teams or families who need to coordinate tasks.

e.Time Management: With to-do apps, users can allocate estimated times for each task, helping them better manage their time and prioritize tasks effectively.

9.Disadvantages of a To-Do App:

a.Learning Curve: For some users, learning how to use a new to-do app can take time, and the initial setup may require some effort.

b.Technical Issues: Like any digital tool, to-do apps can experience technical glitches, server outages, or compatibility issues, disrupting users' ability to access and manage their tasks.

c.Distractions: Constant notifications from a to-do app can also become a distraction, pulling users away from their work or other activities.

d.Dependency on Technology: Relying solely on a digital to-do app might be problematic if the user loses access to their devices or if there's a technical failure.

e.Privacy and Security: Some to-do apps store sensitive personal and work-related information, so users need to be cautious about the app's privacy and security measures.

10.CONCLUSION:

In conclusion, the Todo app stands as an essential tool for individuals seeking to optimize their time and maximize their productivity. By assisting users in organizing, prioritizing, and completing tasks efficiently, it empowers them to lead more balanced and fulfilling lives. With continuous updates and improvements, the Todo app will remain a steadfast companion on the journey towards increased productivity and success.users can easily create, edit, and prioritize tasks, ensuring that nothing slips through the cracks. The inclusion of due dates and reminders will help users stay on top of their commitments and deadlines, reducing the risk of forgetting important responsibilities.