

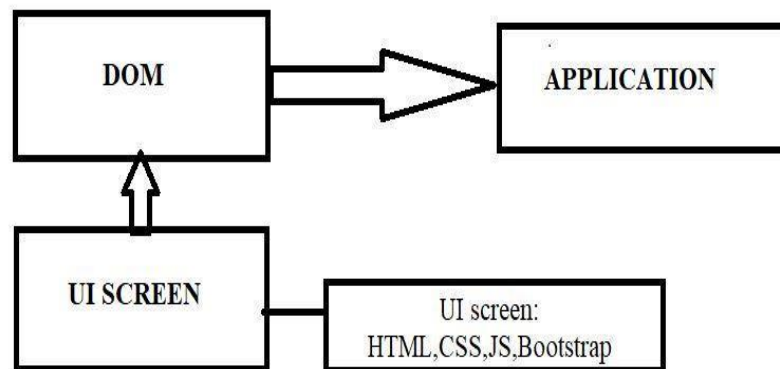
# TODO LIST APP DOCUMENTATION

## INTRODUCTION:

### 1.1 OVERVIEW:

The Todo App is a web application that allows users to create and manage their daily tasks or to-do lists. Users can add new tasks, mark tasks as completed, and delete tasks. The app provides a user-friendly interface and uses JavaScript, HTML, CSS, and Bootstrap for its implementation.

### Technical Architecture



### 1.2 PURPOSE:

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. A to-do list is just a list of things you have to-do. That means basically anything and everything can be on your to-do list—but just

because you've written your to-dos down doesn't mean your to-do list is actually useful. Effectively tracking when your work is due can help you prioritize and get great work done.

## **LITERATURE SURVEY**

### **2.1 Existing Problems:**

#### The Problem with To-Do Lists

It puts pressure on you to finish a specific amount of tasks during the day or week. This can create a huge mental burden — especially if some of the tasks take longer than expected to complete, and you find yourself struggling to get through the list.

### **2.2 Proposed Solutions:**

Creating a proposed solution for a todo list involves understanding the requirements and functionalities needed for the application. Below is a simple outline of a proposed solution for a basic todo list application:

#### **User Interface:**

A clean and intuitive user interface with a list view of tasks.

Option to add, edit, and delete tasks.

Option to mark tasks as completed or incomplete.

Filter and sort options for better task organization.

#### **Backend and Data Storage:**

Server-side backend to handle data storage and retrieval.

A database to store the tasks, preferably with columns for task name, description, status (completed or not), due date, etc.

#### **Functionality:**

Add Task: Allow users to add new tasks to the list.

Edit Task: Users should be able to modify the task details.

Delete Task: Provide the option to remove tasks from the list.

Mark as Completed: Allow users to mark tasks as completed.

Sorting: Allow sorting tasks by due date, completion status, or name.

Filtering: Provide filtering options to show completed or pending tasks.

Search: Implement a search feature to find specific tasks by keywords.

Date/Time Management: Allow users to set due dates and reminders for tasks.

### **User Accounts and Authentication (Optional):**

Implement user accounts to store individual todo lists securely.

User authentication to ensure privacy and data protection.

### **Offline Support (Optional):**

Implement offline support, so users can access and modify their todo list even without an internet connection.

### **Responsive Design (Optional):**

Create a mobile-friendly version for users who prefer using the app on their smartphones or tablets.

### **Data Backup (Optional):**

Provide the option for users to back up their todo list data to avoid accidental loss.

### **Notifications (Optional):**

Add a notification feature to remind users of upcoming due dates or pending tasks.

Collaboration (Optional): Enable users to share their todo lists and collaborate with others on tasks. Always focus on user

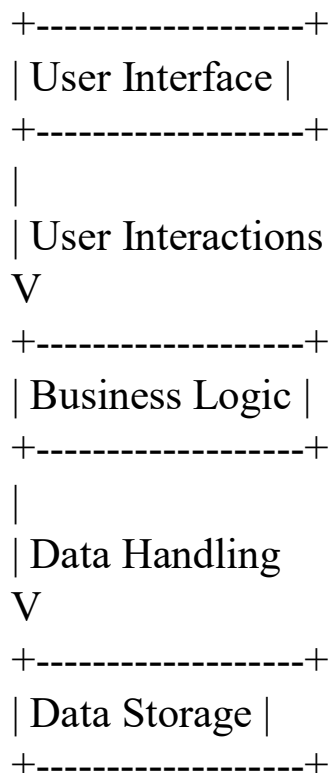
experience, performance, and data security when developing any application.

## **THEORITICAL ANALYSIS:**

### **3.1Block Diagram:**

The block diagram for the front-end development of a TODO App can be represented as follows:

In this block diagram:



Explanation of each block:

#### **Interface:**

This is the front-end layer of the app responsible for presenting the user interface. It displays the task lists, task details, and allows users to interact with the app.

#### **Business Logic:**

The business logic layer contains the core functionalities of the To-Do List app.

It handles task creation, task organization, prioritization, reminders, and notifications.

This layer manages the interactions between the User Interface and the Data Storage layer.

#### **Data Storage:**

The Data Storage layer is responsible for storing and retrieving task-related data.

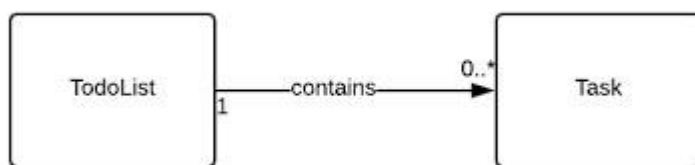
It can use a local database (e.g., SQLite) or cloud-based storage (e.g., Firebase) to store tasks and their attributes.

**The To-Do List app flow would typically work as follows:**

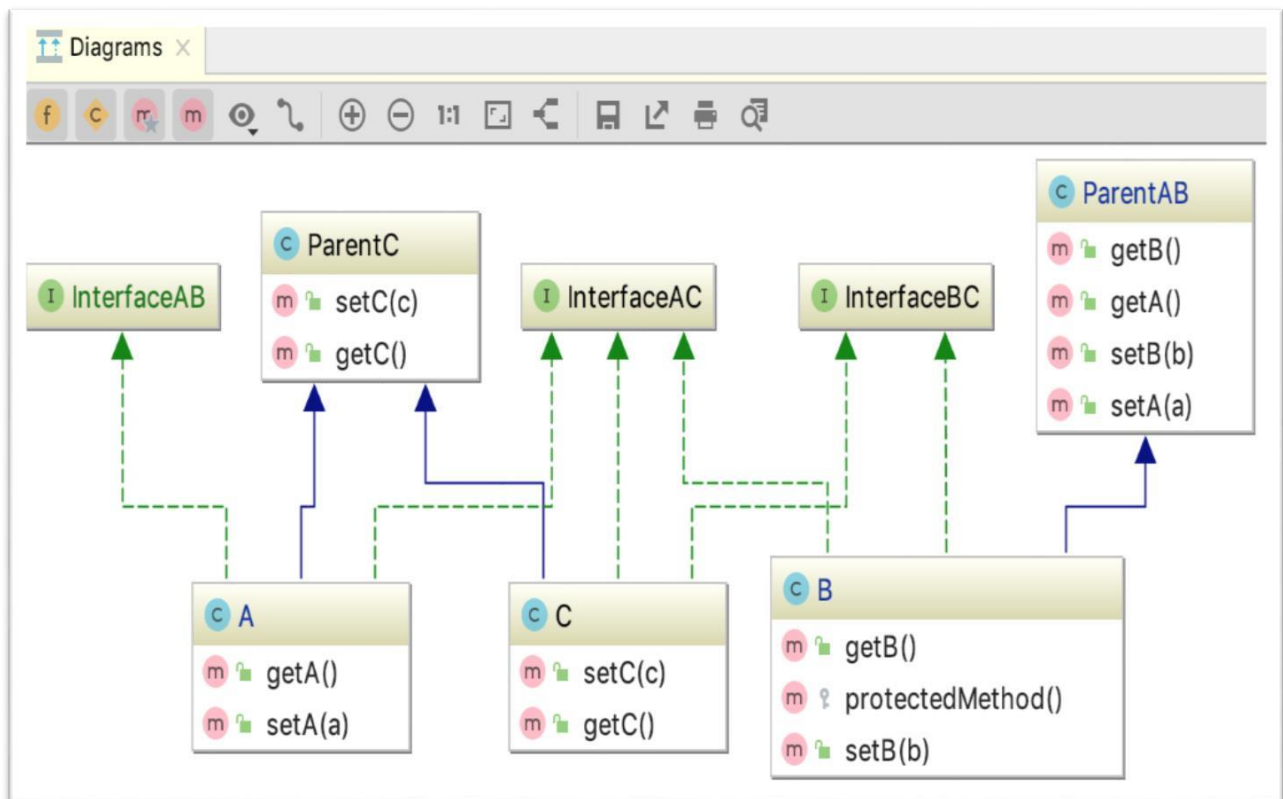
The User Interface displays the task lists and allows users to add, edit, or mark tasks as completed. User interactions and inputs are processed by the Business Logic layer, which handles task management, reminders, and other functionalities.

The Business Logic layer communicates with the Data Storage layer to store and retrieve task data as needed.

The block diagram above provides a high-level overview of the major components involved in a ToDo List app. In practice, there may be additional components and complexities depending on the app's specific features and design



The "User Interface" represents the visual and interactive elements that users will interact with while using the TODO . "TODO App Components" are the front-end components



### 3.2 Hardware / Software designing:

Front-end development focuses on creating the user interface and interactive elements of the TODO App. Below are the hardware and software requirements for front-end development:

#### 3.2.1 Hardware Requirements:

The hardware requirements for front-end development are relatively minimal since most of the development work occurs on personal computers or laptops. However, it's essential to have a capable system with the following specifications:

Computer with sufficient processing power and memory to handle development tools and run modern web browsers efficiently.

High-resolution display for accurate UI design and testing.

#### 3.2.2 Software Requirements:

To build the front-end of the TODO App, developers will need various software tools and frameworks. The specific requirements may vary based on the chosen technology stack, but the **common components include:**

## Text Editor or Integrated Development Environment (IDE):

In this CSS file, we add some styles to improve the appearance of the todo list app. The styles include font styles, input styles, button styles, and list item styles. The tasks are displayed with a line separating each task item, and there is a distinct style for the "Delete" button. This is just one example, and you can further customize the CSS to match your desired design.

Make sure to link this CSS file in your index.html file using the `<link>` tag within the `<head>` section:

```
1  #section2{
2      margin-top: 10vh;
3      padding: 1vh ;
4  }
5  .p {
6      display: flex;
7      align-items: center;
8  }
9
10 svg .bi {
11     margin-right: 5px;
12     /* Adjust the spacing between the icon and text */
13 }
14
15 .important {
16     background-color: #fd826f !important;
17     /* Change the background color to light red */
18 }
19
20
21 #calendar-picker-container , #time-picker-container{
22     width: 50vh;
23 }
24
25 .action-Container{
26     margin: 20px;
27 }
28
29 div .col-md-4{
30     padding: 0vh 3vh;
31     border-radius: 60px 60px 10px 10px;
32     box-shadow: 0px 0px 0px 0.19px 0px 10px 20px, 0px 0px 0px 0.23px 0px 6px 6px;
33 }
34
35 .card1{
36     background-color: rgb(238, 177, 136) !important;
37 }
```

By adding the CSS styles above to your todo list app, it should look much more visually appealing with better formatting and styling for the input, buttons, and task list items. You can always modify and extend the CSS to fit your specific design preferences.

Examples: Visual Studio Code, Sublime Text, Atom, or WebStorm.

Web Development Tools and Frameworks:

```
assets > js > JS script.js > playNotificationSound
1  const cardContainer = document.getElementById('card-container');
2  const tasks = [];
3  const dates = [];
4  const times = [];
5
6  const calendarBtn = document.getElementById('calendar-btn');
7  const calendarPickerContainer = document.getElementById('calendar-picker-container');
8  const calendarSaveBtn = document.getElementById('calendar-save-btn');
9
10 calendarBtn.addEventListener('click', function () {
11     calendarPickerContainer.classList.toggle('d-none');
12 });
13
14 calendarSaveBtn.addEventListener('click', function () {
15     const selectedDate = document.getElementById('calendar-picker').value;
16     if (selectedDate !== '') {
17         dates.push(selectedDate);
18         calendarPickerContainer.classList.add('d-none');
19         saveDataToLocalStorage();
20     }
21 });
22
23 const notificationBtn = document.getElementById('notification-btn');
24 const timePickerContainer = document.getElementById('time-picker-container');
25 const saveBtn = document.getElementById('save-btn');
26
27 notificationBtn.addEventListener('click', function () {
28     timePickerContainer.classList.toggle('d-none');
29 });
30
31 saveBtn.addEventListener('click', function () {
32     const selectedTime = document.getElementById('time-picker').value;
33     if (selectedTime !== '') {
34         times.push(selectedTime);
35         timePickerContainer.classList.add('d-none');
36         saveDataToLocalStorage();
37     }
38 }
```

This script creates a simple todo list where you can enter tasks into an input field, click the "Add Task" button, and the task will be added to the list below. Each task will have a "Delete" button to remove it from the list. The tasks will remain in the list until you manually remove them.

You can customize this code further to add features like storing tasks in local storage, editing tasks, setting due dates, etc., depending on the complexity you want to achieve in your todo list app.



```

40 const taskInput = document.getElementById('task-input');
41 const addBtn = document.getElementById('add-btn');
42
43 taskInput.addEventListener('input', function () {
44   if (taskInput.value.trim() !== '') {
45     addBtn.style.display = 'inline-block';
46   } else {
47     addBtn.style.display = 'none';
48   }
49 });
50
51 addBtn.addEventListener('click', function () {
52   const task = taskInput.value.trim();
53   if (task !== '') {
54     tasks.push({ task: task, date: dates[dates.length - 1], time: times[times.length - 1], notified: false });
55     const card = createCard(task, dates[dates.length - 1], times[times.length - 1]);
56     cardContainer.appendChild(card);
57     taskInput.value = '';
58     addBtn.style.display = 'none';
59     saveDataToLocalStorage();
60     playNotificationSound(); // Play the notification sound on adding a task
61   }
62 });
63
64 document.addEventListener('DOMContentLoaded', function () {
65   loadDataFromLocalStorage();
66   checkDateTime();
67   setInterval(checkDateTime, 1000);
68 });
69
70 function createCard(task, date, time) {
71   const card = document.createElement('div');
72   card.classList.add('card', 'col-md-4', 'my-3');
73
74   const cardBody = document.createElement('div');
75   cardBody.classList.add('card-body');
76

```

I apologize for the confusion, but it seems you have a typo in your request. However, based on the context, I assume you meant "index.html." Here's the updated and complete index.html file for the todo list app, including the CSS styles:

This example includes the HTML structure for the todo list app, CSS styles to make it visually appealing, and JavaScript code to handle adding tasks and deleting tasks from the list.

Ensure you save the CSS styles in "style.css" and the JavaScript code in "script.js" files in the same directory as the index.html file. The CSS styles will be applied to the elements in the HTML file, and the JavaScript code will handle the todo list functionality, allowing you to add tasks and delete tasks from the list.

This example includes the HTML structure for the todo list app, CSS styles to make it visually appealing, and JavaScript code to handle adding tasks and deleting tasks from the list.

Ensure you save the CSS styles in "style.css" and the JavaScript code in "script.js" files in the same directory as the index.html file. The CSS styles will be applied to the elements in the HTML file, and the JavaScript code will handle the todo list functionality, allowing you to add tasks and delete tasks from the list.

```

1 <!doctype html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>ToDo List</title>
8   <link rel="stylesheet" href="./assets/css/style.css">
9
10  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
11    integrity="sha384-9ndCyuaIbZAI2FUVXJi0CjmCapSm07SnpJef0486ghLnuZ2cderh002iuK6FUUVM" crossorigin="anonymous">
12  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
13 </head>
14
15 <body>
16   <section id="section1">
17
18     <nav class="navbar bg-transparent fixed-top">
19       <div class="container-fluid">
20         <button class="navbar-toggler me-auto" type="button" data-bs-toggle="offcanvas"
21           data-bs-target="#offcanvasNavbar" aria-controls="offcanvasNavbar">
22           <span class="navbar-toggler-icon"></span>
23         </button>
24         <div class="offcanvas offcanvas-start tabindex="-1" id="offcanvasNavbar"
25           aria-labelledby="offcanvasNavbarLabel">
26           <div class="offcanvas-header">
27             <h5 class="offcanvas-title" id="offcanvasNavbarLabel">TODO APP</h5>
28             <button type="button" class="btn-close" data-bs-dismiss="offcanvas"
29               aria-label="Close"></button>
30           </div>
31           <div class="offcanvas-body">
32             <ul class="navbar-nav justify-content-start flex-grow-1 pe-3">
33
34               <li class="nav-item">
35                 <a class="nav-link active" aria-current="page" href="index.html">
36                   <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
37                     fill="currentColor" class="bi bi-list-task" viewBox="0 0 16 16">

```

```

               <path
38                 d="M1.5 2.5A.5.5 0 0 1 2 2h12a.5.5 0 0 1 0 1h-12a.5.5 0 0 1-.5-.5zm0 4A.5.5 0 0 1 2 6h12a.5.5
39               </svg>
40               Task
41             </a>
42           </li>
43           <!-- <li class="nav-item">
44             <a class="nav-link" href="#">
45               <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
46                 fill="currentColor" class="bi bi-calendar" viewBox="0 0 16 16">
47                 <path
48                   d="M13 2h-1v1a1 1 0 0 0 1 1h-2a1 1 0 0 0 1 1v1H6V1a1 1 0 0 0 1 1h3a1 1 0 0 0 1 1v1H1a1 1 0 0
49               </svg>
50               My Day
51             </a>
52           </li>
53           <li class="nav-item">
54             <a class="nav-link" href="#">
55               <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
56                 fill="currentColor" class="bi bi-exclamation-triangle"
57                 viewBox="0 0 16 16">
58                 <path
59                   d="M8.252 0.275a1.063 1.063 0 0 1 1.496 0l6.98 6.77A1.063 1.063 0 0 1 16 8.464v5.274c0 .587-.
60               </svg>
61               Important
62             </a>
63           </li>
64           <li class="nav-item">
65             <a class="nav-link" href="#">
66               <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
67                 fill="currentColor" class="bi bi-clock" viewBox="0 0 16 16">
68                 <path
69                   d="M7.5 1a6.5 6.5 0 0 0 0 6.496 6.499L1 8.5a.5.5 0 0 0 .55h1.645a.5.5 0 0 0 .415-.777L3.8 7.3A
70                 <path
71                   d="M7.5 3a.5.5 0 0 0-.5.5v4.085l3.25 1.925a.5.5 0 0 0 .5-.866L7.5 7.707V3.5a.5.5 0 0 0 7.5 3
72               </svg>
73             Planned

```

**HTML (Hypertext Markup Language):** To structure the content and layout of the app.

**CSS (Cascading Style Sheets):** To style the user interface and define the app's appearance.

JavaScript: To implement interactivity, handle user events, and manage the app's behavior. **Front-end Frameworks and Libraries:** Popular choices include React.js, Angular, Vue.js, or other JavaScript frameworks. UI/UX Design Tools (optional):

Tools like Sketch, Adobe XD, Figma, or InVision may be used for creating UI mockups and prototypes. **Version Control:**

Git: To track changes in the codebase and collaborate with other developers. **Web Browsers:**

Chrome, Firefox, Safari, or other modern web browsers for testing and debugging. **Package Managers:**

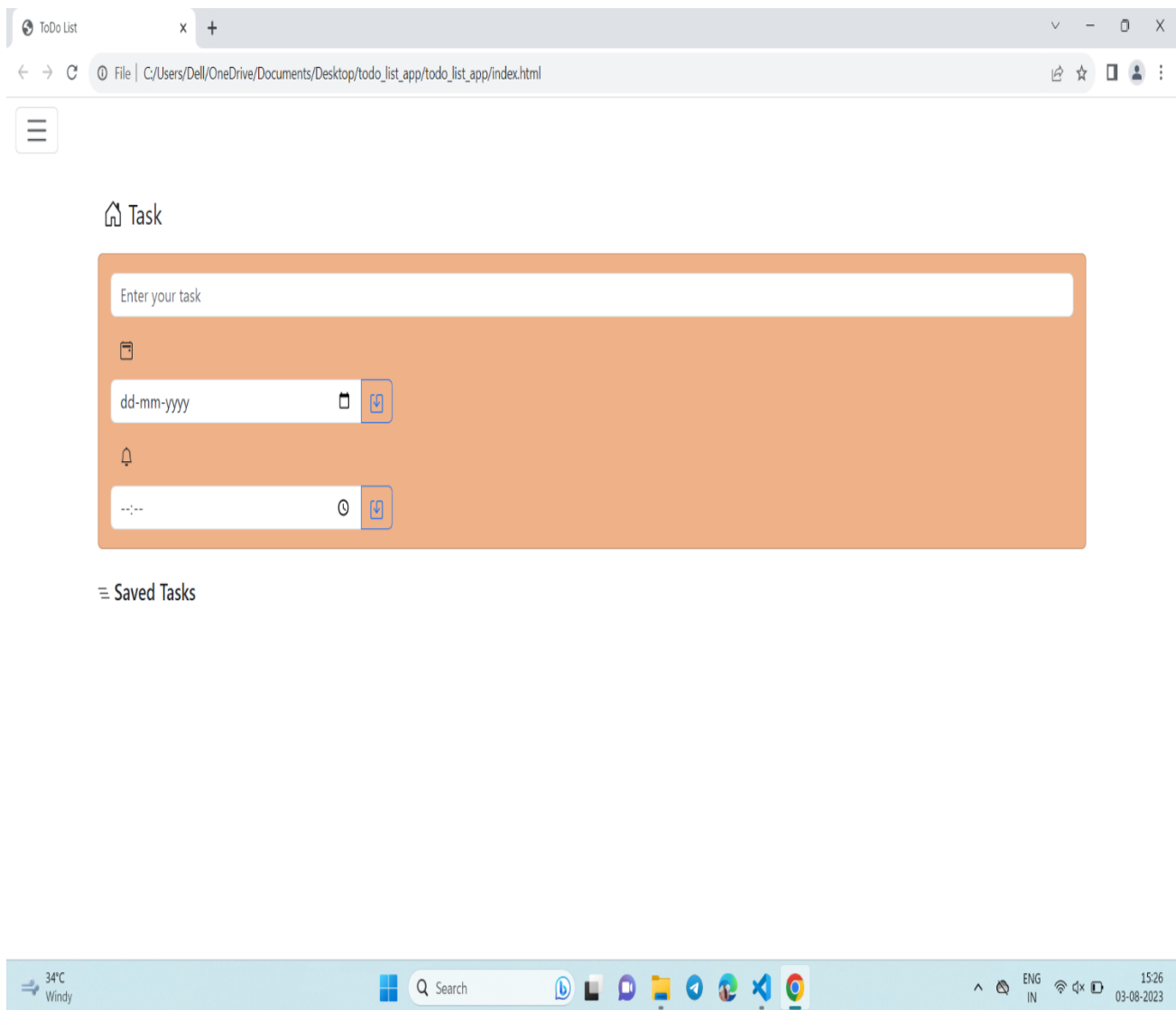
NPM (Node Package Manager) or Yarn for managing JavaScript dependencies.

### **Build Tools:**

Webpack, Parcel, or other build tools to bundle and optimize the front-end code.

It's important to note that front-end development for a TODO App may involve integrating with back-end APIs (e.g., for data storage or user authentication). Therefore, front-end developers should also be familiar with making API requests and handling responses using technologies like Fetch API, Axios, or other AJAX libraries. The specific choice of front-end tools and frameworks may depend on the developer's familiarity, project requirements, and team preferences.

## RESULT



After given the input






**127.0.0.1:5500 says**  
Task notification111  
**OK**

 **Task**


 



### Saved Tasks

**task**  
Date: 2023-08-03  
Time: 22:43  
 





**127.0.0.1:5500 says**  
ravi task  
**OK**

 **Task**

### Saved Tasks

**after chancing  
task**  
Date: 2023-08-03  
Time: 23:05  
 

## ADVANTAGES AND DISADVANTAGES

### **Advantages:**

**Enhanced user experience:** A well-designed frontend will provide a seamless and intuitive user interface, leading to a positive user experience and increased user engagement.

**Responsive design:** Developing a responsive frontend ensures that the Todo app works well on various devices and screen sizes, improving accessibility and usability for users.

**Real-time updates:** Implementing real-time updates using technologies like WebSocket or WebRTC can enable users to see changes made by collaborators instantly, promoting collaboration and efficiency.

**Faster load times:** Optimized frontend development techniques can result in faster loading times, reducing user frustration and encouraging them to use the app more frequently.

**Cross-platform compatibility:** By using frameworks like React Native or Flutter, the frontend can be adapted for both web and mobile platforms, reaching a wider audience.

### **Disadvantages :**

**Technical complexity:** Frontend development can become complex, especially when integrating with backend systems, APIs, or third-party services, potentially leading to bugs and maintenance challenges.

**Browser compatibility issues:** Ensuring that the app works seamlessly across different browsers and versions can be challenging, requiring additional testing and compatibility fixes.

**Performance bottlenecks:** Overloading the frontend with heavy animations or excessive data can lead to performance issues, impacting the app's responsiveness and user experience.

**Security vulnerabilities:** Inadequate security measures in the frontend code might expose the app to potential attacks like cross-site scripting (XSS) or injection attacks, compromising user data.

**Version compatibility:** As technology evolves, new updates and versions of frontend libraries or frameworks may be released,

necessitating continuous maintenance and updates to keep the app current and secure.

It's crucial to have a skilled and experienced frontend development team to mitigate these potential disadvantages and ensure a successful implementation of the proposed Todo app frontend. Regular testing, security audits, and optimization efforts are also vital to overcome challenges and provide a robust and user-friendly frontend experience.

## **APPLICATIONS:**

The Todo app can be applied in various areas and industries to improve task management and organization. Here are some potential applications:

**Personal Productivity:** Individuals can use the Todo app to manage their daily tasks, set reminders, and prioritize activities, leading to increased personal productivity and time management.

**Project Management:** Teams working on projects can utilize the Todo app to assign tasks, track progress, and ensure that deadlines are met, facilitating effective project management. **Task**

**Collaboration:** In collaborative environments, such as workplaces or group projects, the Todo app can enable team members to share tasks, comment on updates, and coordinate efforts efficiently.

**Education:** Students can use the Todo app to keep track of assignments, study schedules, and extracurricular activities, aiding in academic organization and time management.

**Personal Goal Setting:** The Todo app can be employed as a tool to set and track personal goals, whether related to fitness, learning new skills, or pursuing hobbies.

**Event Planning:** Organizers can use the Todo app to create checklists for event planning, including tasks like sending invitations, arranging venues, and coordinating logistics.

**Household Chores:** Families can benefit from using the Todo app to delegate household chores, ensuring tasks are evenly distributed and completed on time.

**Travel Planning:** Travelers can organize their trip by creating lists of things to pack, places to visit, and activities to do during their journey.

**Time Blocking:** Professionals or students can implement time blocking techniques using the Todo app to allocate specific time slots for different tasks throughout the day.

**Business Workflow:** Businesses can integrate the Todo app into their workflow to streamline daily operations, manage team tasks, and improve overall productivity.

The versatility of the Todo app makes it applicable in various contexts, making task management more efficient and organized across personal, professional, and educational settings.

## CONCLUSION:

App *todo-list-app* performs very efficient comparing to competitor. There is a space for further developments with regards of keeping app small and quick. It would be optimal to use dedicated CSS and continuing using vanilla Java Script. To-do-app can be developed as a sole application as well as a very efficient module to be combined in a larger project. One of the key challenges is to choose appropriate storage solution, that will allow to maintain its biggest advantages.

- simplicity
- speed
- low recourses demand

## FUTURE SCOPE

The future scope of enhancements for the Todo app frontend development is vast, and continuous improvements can ensure the app remains relevant and user-friendly. Here are eight potential enhancements that can be made:

**Advanced Collaboration Features:** Implementing more advanced collaboration features, such as real-time task editing,



chat functionality, and task assignment, can enhance teamwork and streamline project management.

**Intelligent Task Prioritization:** Introducing AI-powered algorithms to help users prioritize tasks based on deadlines, importance, and user behavior can make the Todo app even more personalized and efficient.

**Voice Integration:** Integrating voice commands and voice-to-text functionality can allow users to add tasks or update their lists hands-free, making the app more accessible and user-friendly.

**Smart Notifications:** Utilize smart notifications that adapt to users' preferences and habits, ensuring timely reminders without overwhelming users with unnecessary alerts.

**Gamification:** Adding gamification elements, such as rewards, badges, or progress tracking, can boost user engagement and motivation to complete tasks, turning task management into a more enjoyable experience.

**Dark Mode:** Implementing a dark mode option can enhance user experience and reduce eye strain, especially during nighttime usage or in low-light environments.

**Integration with Other Apps:** Allowing integration with other productivity apps, calendars, or project management tools can streamline workflows and centralize task management across multiple platforms.

**Data Analytics and Insights:** Providing users with analytics and insights into their task completion patterns, productivity trends, and time allocation can empower them to make data driven decisions and optimize their productivity.

Overall, the future scope of the Todo app frontend development involves leveraging emerging technologies, user feedback, and industry trends to create a more feature-rich, intuitive, and personalized user experience. Continuous updates, improvements, and innovation are essential to stay ahead in the competitive market and ensure the Todo app remains a valuable tool for task organization and productivity enhancement.