MASTERING TASK MANAGEMENT THE ULTIMATE SAVEETHA TO-DO LIST GUIDE

A CAPSTONE PROJECT

Submitted By

SAI MAHESH G (192211122)

AKHIL G (192211009)

In Partial Fulfillment for the completion of the course

CSA0910

PROGRAMMING IN JAVA FOR WEB APPLICATION



SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CHENNAI - 602105 TAMIL NADU, INDIA



# BONAFIDE CERTIFICATE

This is to certify that the project report entitled **MASTERING TASK MANAGEMENT: THE ULTIMATE SAVEETHA TO-DO LIST GUIDE** submitted by G.Sai Mahesh 192211122 and G. Akhil 192211009, to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by him/her under my guidance. The project fulfills the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

Dr.A.G.Ramachandhran

**COURSE FACULTY**

*Department of Blockchain and cybersecurity. Saveetha School of Engineering, SIMATS, Chennai - 602105*

# ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N M VEERAIYAN**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. DEEPAK NALLASWAMY,** SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. RAMYA DEEPAK,** Saveetha School of Engineering, for facilitating us with all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B RAMESH,** Principal, Saveetha School of Engineering for my support in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty Dr.A.G.Ramachandhran, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire Department of Design faculty for their constructive criticisms and valuable suggestions, which have been a rich source to improve the quality of this work.

**INDEX**

[BONAFIDE CERTIFICATE](#_bookmark0) 1

[ACKNOWLEDGEMENT](#_bookmark1) 2

1. [ABSTRACT](#_bookmark2) 4
2. [INTRODUCTION](#_bookmark3) 5
3. [ARCHITECTURE DIAGRAM](#_bookmark4) 6
4. [FLOWCHART](#_bookmark5) 8
5. [UML DIAGRAM](#_bookmark6) 10
6. [CLASS DIAGRAM](#_bookmark6) 10
7. [CODE IMPLEMENTATION](#_bookmark7) 12
   1. [JAVA CODE](#_bookmark8) 12
   2. [HTML CODE](#_bookmark9) 13
   3. [CSS CODE](#_bookmark10) 13
8. [OUTPUT SCREENSHOT](#_bookmark11) 18
9. [CONCLUSION](#_bookmark12) 21
10. [REFERENCES](#_bookmark13) 23

# ABSTRACT

This capstone project presents the To-Do List application serves as an essential tool for

personal productivity using Java for enterprise applications. The project involves the inclusion

of date, time, and timed reminders allows users to set deadlines and receive notifications,

improving time management and accountability. The design’s use of modern, dynamic styling,

such as CSS animations and live backgrounds, ensures aesthetic appeal and usability. This

functionality is ideal for individuals seeking to streamline personal and professional

responsibilities while ensuring flexibility in a busy lifestyle.

# INTRODUCTION

A Saveetha To-Do List is a fundamental tool designed to help individuals and teams manage tasks and responsibilities more effectively. By organizing tasks into a list format, users can prioritize actions, monitor progress, and ensure important activities are not overlooked. The application simplifies task management by allowing users to create, modify, and track tasks with ease.

To-do lists are increasingly adopted in both personal and professional settings due to their flexibility, providing users with a structured way to handle deadlines, appointments, and goals. With modern enhancements such as mobile responsiveness, task counting, reminders, and colorful designs, to-do lists have evolved into highly interactive and engaging platforms for productivity.

In recent times, digital to-do lists have been enhanced with several advanced features. They support real-time syncing across multiple devices, ensuring users can access their tasks anywhere, anytime. Some even incorporate gamification elements, making task completion more engaging. Integrating features such as timers, pop-up reminders, and notifications further aids in time management.

A To-Do List application is essential for personal productivity and team collaboration, offering a simple yet effective way to organize tasks and manage time. It provides a clear structure for handling daily responsibilities, helping users break down complex activities into manageable items. By categorizing tasks based on priority, urgency, or context, users can focus on what matters most while minimizing distractions. This improves efficiency, reduces stress, and promotes accountability.

Additionally, the modern to-do list is increasingly integrated with AI-powered suggestions, which can help predict deadlines, prioritize tasks, or even automate recurring tasks based on previous patterns. As a result, to-do lists are transforming into comprehensive task management solutions, driving productivity and fostering better time management in both personal and professional environments.

# ARCHITECTURE DIAGRAM

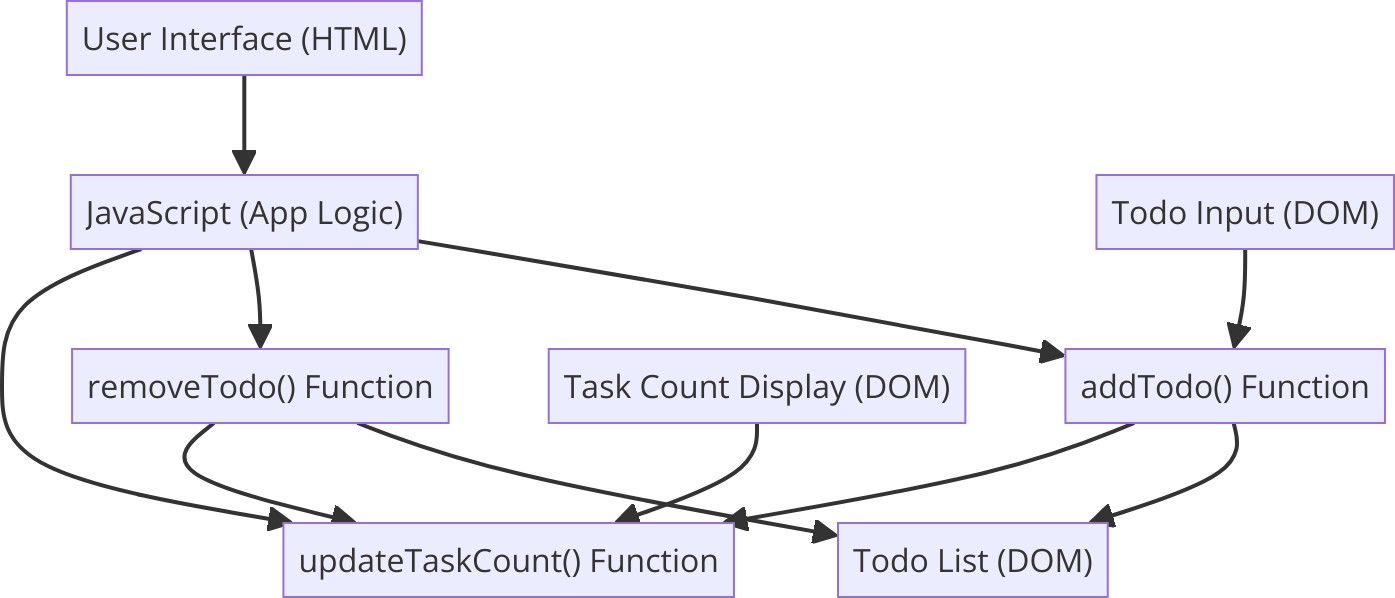


Figure 3.1 : Architecture Diagram

The architecture diagram of the to-do list application illustrates the various components and their interactions within the system. This modular architecture ensures scalability, maintainability, and efficient performance.

* **Overall Structure:** The architecture diagram represents the high-level design of the to-do list application.
* **Components:** It consists of three main layers: Presentation Layer, Business Logic Layer, and Data Layer.
* **Presentation Layer:** This layer includes the user interface components built with HTML, CSS, and JavaScript. It handles user interactions and displays the to-do list, including tasks, due dates, and timers.
* **Business Logic Layer:** Implemented in Java, this layer manages the core functionalities of the to-do list, such as task management, timer logic, and notifications.
* **Data Layer:**Responsible for managing the to-do list data, which can be stored locally or fetched from a cloud database**.**
* **Communication:** The layers interact with each other using defined protocols, ensuring modularity and maintainability.

This diagram delineates its modular structure, emphasizing the separation of concerns to ensure maintainability and scalability. The system is divided into three primary layers: the Presentation Layer, the Business Logic Layer, and the Data Layer.

The Presentation Layer forms the interface that users interact with. Built using HTML for structure, CSS for styling, and JavaScript for interactivity, it allows users to add, update, and remove tasks, set deadlines, and utilize the timer feature. This layer ensures a smooth and intuitive user experience, providing immediate feedback and displaying task details clearly.

The Business Logic Layer, developed in Java, is the core of the application where the main task management and timer logic reside. It processes user inputs from the Presentation Layer, applies the necessary operations, and manages notifications and updates. This layer also

handles validation to ensure accurate task management and timer functionality.

The Data Layer is responsible for managing the to-do list data. It can either use local storage for simplicity or fetch data from a cloud-based database for enhanced accessibility. This layer ensures that the to-do list data is reliably stored and synchronized. The communication between these layers follows a clear protocol, ensuring that each layer performs its function independently while contributing to the overall workflow of the application.

# FLOWCHART

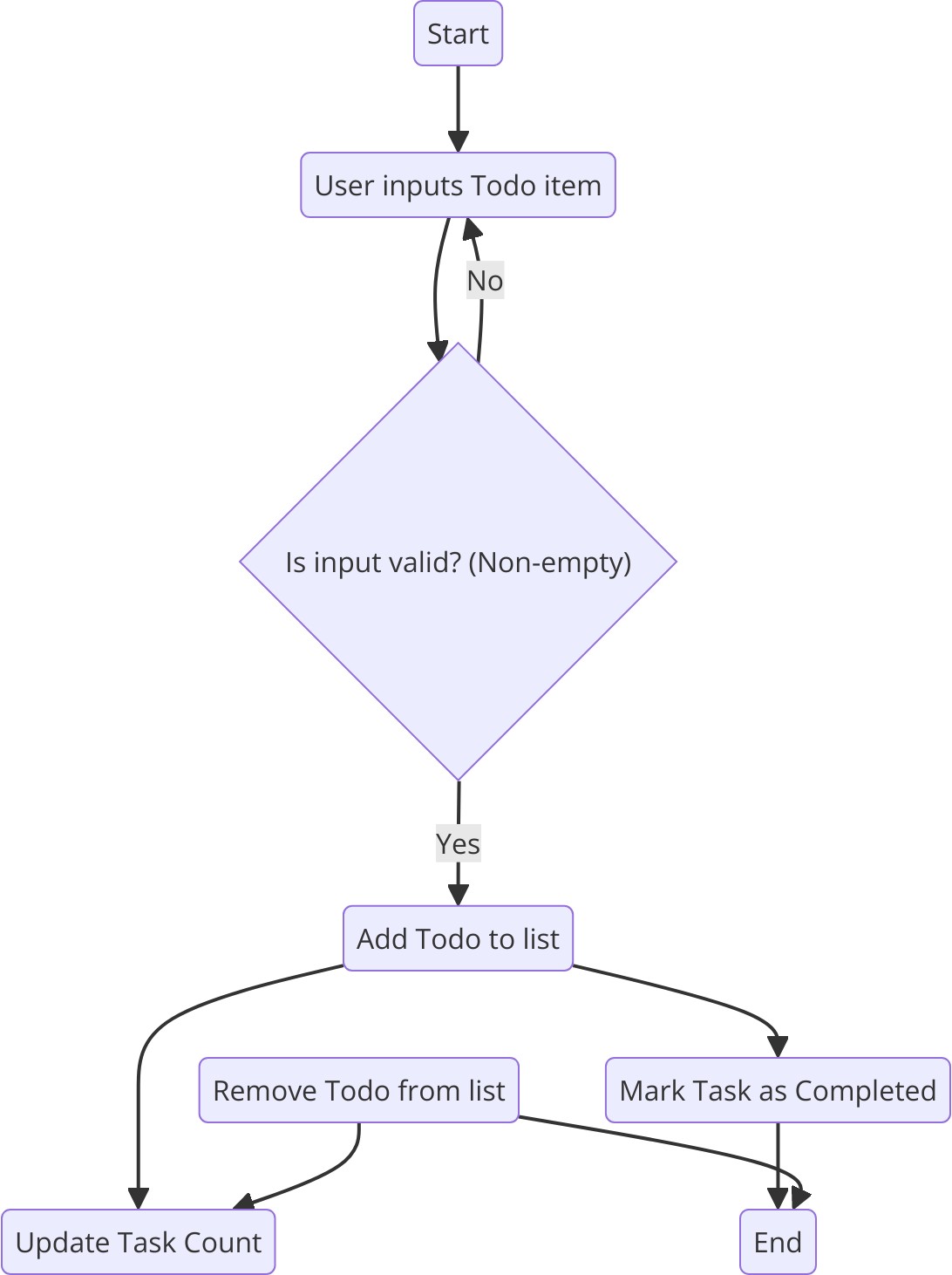


Figure 4.1 : Flowchart

The flowchart for the to-do list application depicts the step-by-step process from task creation to task completion and management.

* **Start:** The process begins when the user accesses the to-do list application.
* **Input:** User inputs a new task, sets a due date, and optionally specifies a timer.
* **Validation:** The system checks if the task details are valid (non-empty and within allowed date range).
* **Fetch Rates:** The system stores the task information in the data layer.
* **Update Display:** The to-do list is updated to reflect the newly added or modified task.
* **Manage Tasks:** Users can update, delete, or mark tasks as completed.
* **Timer Notification:** If a timer is set, the system triggers notifications or reminders as needed.
* **End:** The process completes, and the user can continue managing their tasks or exit the application.

This diagram visually represents the sequential steps involved in managing tasks within the application. It starts with the user accessing the interface, where they can input new tasks, set due dates, and optionally configure a timer.

Once the user inputs the task details, the system performs a validation check to ensure the task information is valid. This step is crucial to prevent errors and ensure that tasks are accurately stored and managed. If the task details are invalid, the user is prompted to correct the information.

Upon successful validation, the system saves the task information to the data layer. This step ensures that the task is stored securely and can be accessed later. The to-do list is then updated to reflect the new or modified task, providing immediate feedback to the user.

Users can manage their tasks by updating, deleting, or marking them as completed. If a timer is set, the system manages notifications or reminders based on the timer settings. The process ensures a smooth and efficient flow from task creation to management, making the application user-friendly and effective for personal organization.

# CLASS DIAGRAM

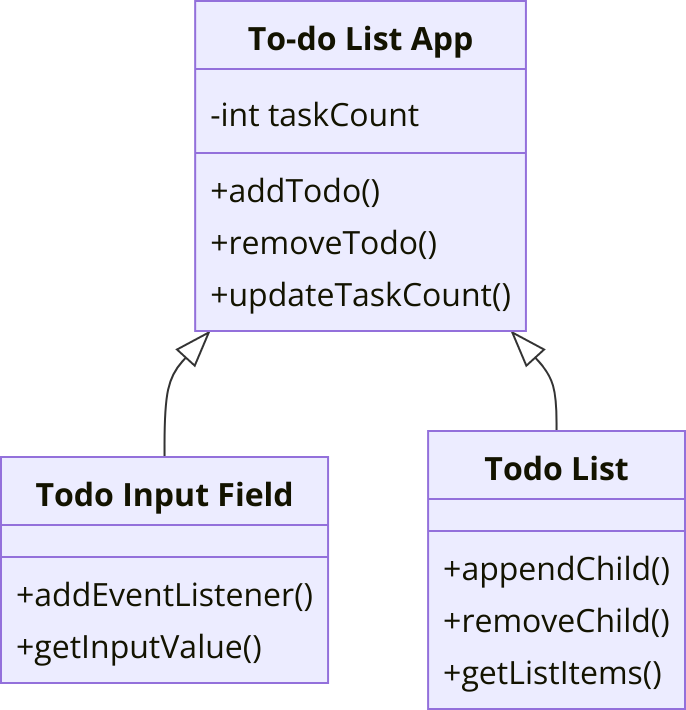


Figure 6.1 : Class Diagram

The class diagram specifies the structure of the to-do list application by showing the system's classes, their attributes, methods, and the relationships among the objects.

**Attributes and Methods:** Detailed attributes and methods for each class.

**Task:** Attributes include taskName, dueDate, and isCompleted; methods include addTask(), updateTask(), and deleteTask().

**Timer**: Attributes include timerDuration and isActive; methods include startTimer(), stopTimer(), and checkTimer().

**TaskManager:** Attributes include taskList; methods include getTasks(), addTask(), removeTask(), and updateTask().

**UserInterface:** Attributes include taskInputField, dueDateSelector, and taskListDisplay; methods include getUserInput(), displayTasks(), and showNotification().

This diagram provides an in-depth look at the attributes and methods of each class, illustrating how they interact to manage tasks. The main classes in the diagram are Task, Timer, TaskManager, and UserInterface.

The Task class includes attributes such as taskName, dueDate, and isCompleted, which represent the details of a task. Its methods, such as addTask(), updateTask(), and deleteTask(), manage the lifecycle of individual tasks.

The Timer class manages task-related timers. Attributes like timerDuration and isActive track the timer settings, while methods like startTimer(), stopTimer(), and checkTimer() handle

timer operations and notifications.

The TaskManager class is responsible for managing the overall list of tasks. Its attribute taskList holds all tasks, and its methods like getTasks(), addTask(), removeTask(), and updateTask() handle the creation, retrieval, deletion, and updating of tasks.

The UserInterface class manages interactions with the user. Attributes such as taskInputField, dueDateSelector, and taskListDisplay capture and display user inputs. Methods like getUserInput(), displayTasks(), and showNotification() handle user input and provide feedback.

Interactions between these classes ensure a seamless workflow. For example, the TaskManager class uses methods from the Task and Timer classes to update and manage tasks. The UserInterface class interacts with TaskManager to display tasks and update the user on timer notifications. This collaboration between classes ensures a smooth and efficient task management experience..

# CODE IMPLEMENTATION

## JAVA CODE

import java.util.ArrayList;

import java.util.Scanner;

class Task {

private String description;

private boolean isCompleted;

public Task(String description) {

this.description = description;

this.isCompleted = false;

}

public String getDescription() {

return description;

}

public boolean isCompleted() {

return isCompleted;

}

public void markAsCompleted() {

this.isCompleted = true;

}

@Override

public String toString() {

return (isCompleted ? "[X] " : "[ ] ") + description;

}

}

class ToDoList {

private ArrayList<Task> tasks;

public ToDoList() {

tasks = new ArrayList<>();

}

public void addTask(String description) {

tasks.add(new Task(description));

}

public void removeTask(int index) {

if (index >= 0 && index < tasks.size()) {

tasks.remove(index);

} else {

System.out.println("Invalid task index.");

}

}

public void markTaskAsCompleted(int index) {

if (index >= 0 && index < tasks.size()) {

tasks.get(index).markAsCompleted();

} else {

System.out.println("Invalid task index.");

}

}

public void displayTasks() {

if (tasks.isEmpty()) {

System.out.println("Your to-do list is empty.");

} else {

for (int i = 0; i < tasks.size(); i++) {

System.out.println(i + ". " + tasks.get(i));

}

}

}

}

public class ToDoListApp {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

ToDoList toDoList = new ToDoList();

boolean running = true;

while (running) {

System.out.println("\nTo-Do List:");

toDoList.displayTasks();

System.out.println("\nOptions: ");

System.out.println("1. Add a task");

System.out.println("2. Remove a task");

System.out.println("3. Mark a task as completed");

System.out.println("4. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

System.out.print("Enter task description: ");

String description = scanner.nextLine();

toDoList.addTask(description);

break;

case 2:

System.out.print("Enter task index to remove: ");

int removeIndex = scanner.nextInt();

toDoList.removeTask(removeIndex);

break;

case 3:

System.out.print("Enter task index to mark as completed: ");

int completeIndex = scanner.nextInt();

toDoList.markTaskAsCompleted(completeIndex);

break;

case 4:

running = false;

break;

default:

System.out.println("Invalid option. Try again.");

}

}

System.out.println("Goodbye!");

scanner.close();

}

}

## HTML CODE

<body>

<div class="container">

<h1>SAVEETHA To-Do List</h1>

<div class="input-area">

<input type="text" id="todo-input" placeholder="Add a new task...">

<button id="add-btn" onclick="addTodo()">Add Task</button>

</div>

<ul id="todo-list"></ul>

<div class="counter">

<p>Total Tasks: <span id="task-count">0</span></p>

</div>

</div>

<!-- Font Awesome for icons -->

<script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous"></script>

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

</body>

## CSS CODE

/\* Import Google Fonts \*/ @import

url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap');

/\* General Styles \*/ body {

font-family: 'Poppins', sans-serif;

background: linear-gradient(to right, #FFDEE9, #B5FFFC); /\* Soft Gradient Background \*/ margin: 0;

padding: 0; display: flex;

justify-content: center;

align-items: center; height: 100vh; color: #333;

}

.container {

background-color: rgba(255, 255, 255, 0.85); /\* Semi-transparent white \*/ padding: 30px 20px;

border-radius: 15px;

box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);

width: 90%;

max-width: 500px; text-align: center;

}

h1 {

font-size: 2.5em; margin-bottom: 25px;

color: #1E90FF; /\* Vibrant Blue \*/

}

.input-area { display: flex;

justify-content: space-between; margin-bottom: 25px;

}

#todo-input { width: 70%;

padding: 12px 15px; font-size: 16px; border-radius: 30px;

border: 2px solid #1E90FF; outline: none;

transition: border-color 0.3s ease;

}

#todo-input:focus {

border-color: #FF69B4; /\* Bright Pink on focus \*/

}

#add-btn {

padding: 12px 20px; font-size: 16px;

border-radius: 30px; border: none;

background-color: #1E90FF; /\* Vibrant Blue \*/ color: white;

cursor: pointer;

transition: background-color 0.3s ease, transform 0.2s ease; display: flex;

align-items: center; justify-content: center;

}

#add-btn:hover {

background-color: #FF69B4; /\* Bright Pink on hover \*/ transform: scale(1.05);

}

#add-btn i { margin-left: 8px;

}

#todo-list {

list-style-type: none; padding: 0;

margin: 0;

text-align: left;

}

#todo-list li {

background-color: #FFF; margin: 10px 0;

padding: 15px 20px; border-radius: 30px; display: flex;

justify-content: space-between; align-items: center;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

transition: transform 0.2s ease, background-color 0.3s ease;

}

#todo-list li:hover { transform: translateY(-3px);

background-color: #FFDEE9; /\* Light Pink \*/

}

#todo-list li.completed {

text-decoration: line-through; opacity: 0.6;

}

#todo-list li button { border: none;

background-color: #FF6347; /\* Tomato Red \*/ color: white;

padding: 8px 12px; border-radius: 20px; cursor: pointer;

transition: background-color 0.3s ease, transform 0.2s ease; display: flex;

align-items: center; justify-content: center;

}

#todo-list li button:hover {

background-color: #FF4500; /\* Orange Red on hover \*/ transform: scale(1.1);

}

.counter {

margin-top: 25px; font-size: 1.3em;

color: #32CD32; /\* Fresh Green \*/

}

.counter span {

font-weight: bold;

color: #1E90FF; /\* Vibrant Blue \*/

}

/\* Responsive Design \*/ @media (max-width: 600px) {

.container {

padding: 25px 15px;

}

h1 {

font-size: 2em;

}

#todo-input { width: 65%;

}

#add-btn {

padding: 10px 15px;

}

#todo-list li { padding: 12px 15px;

}

.counter {

font-size: 1.1em;

}

}

# OUTPUT SCREENSHOT

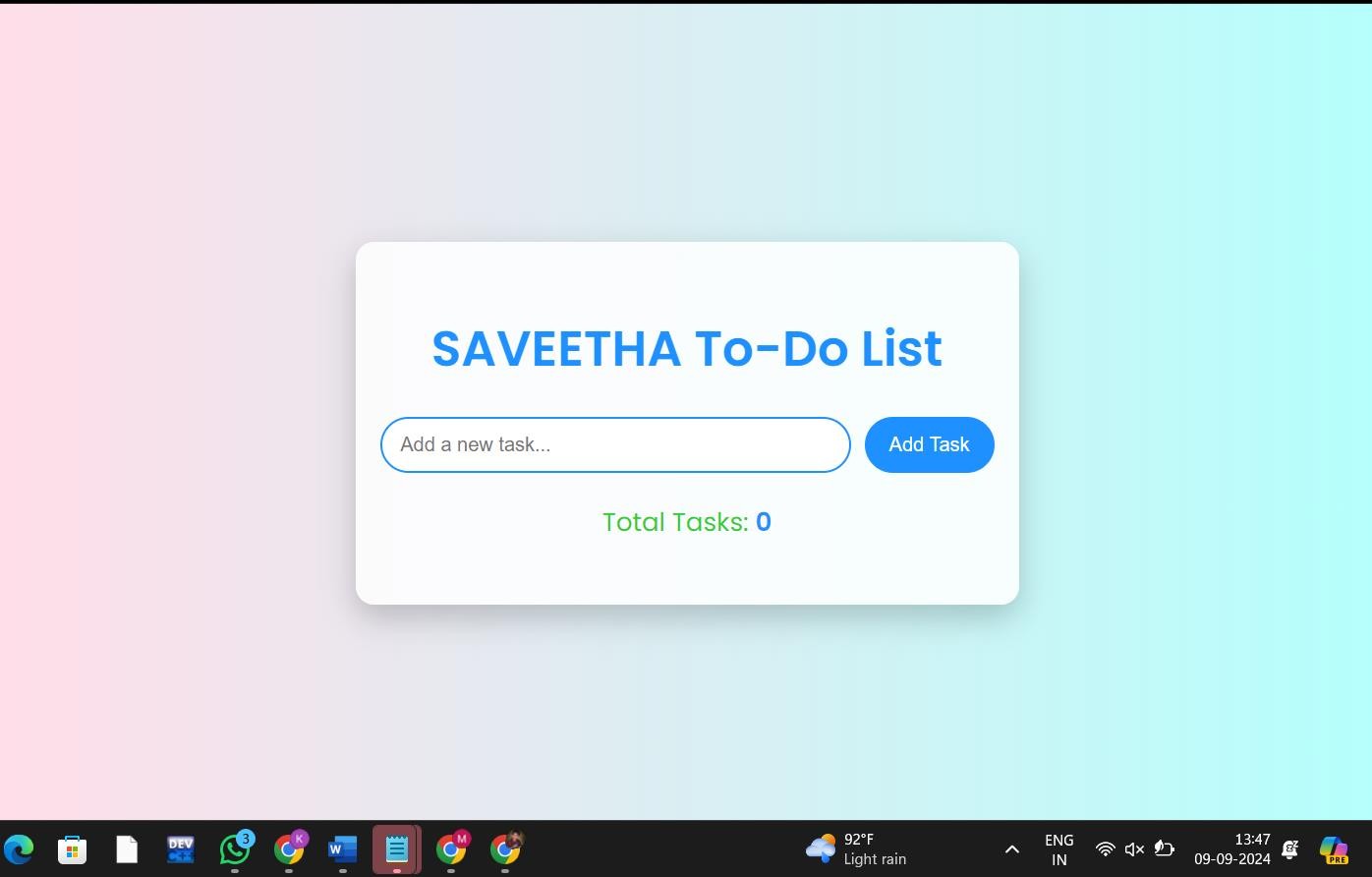


Figure 8.1 : Opening the To-Do List

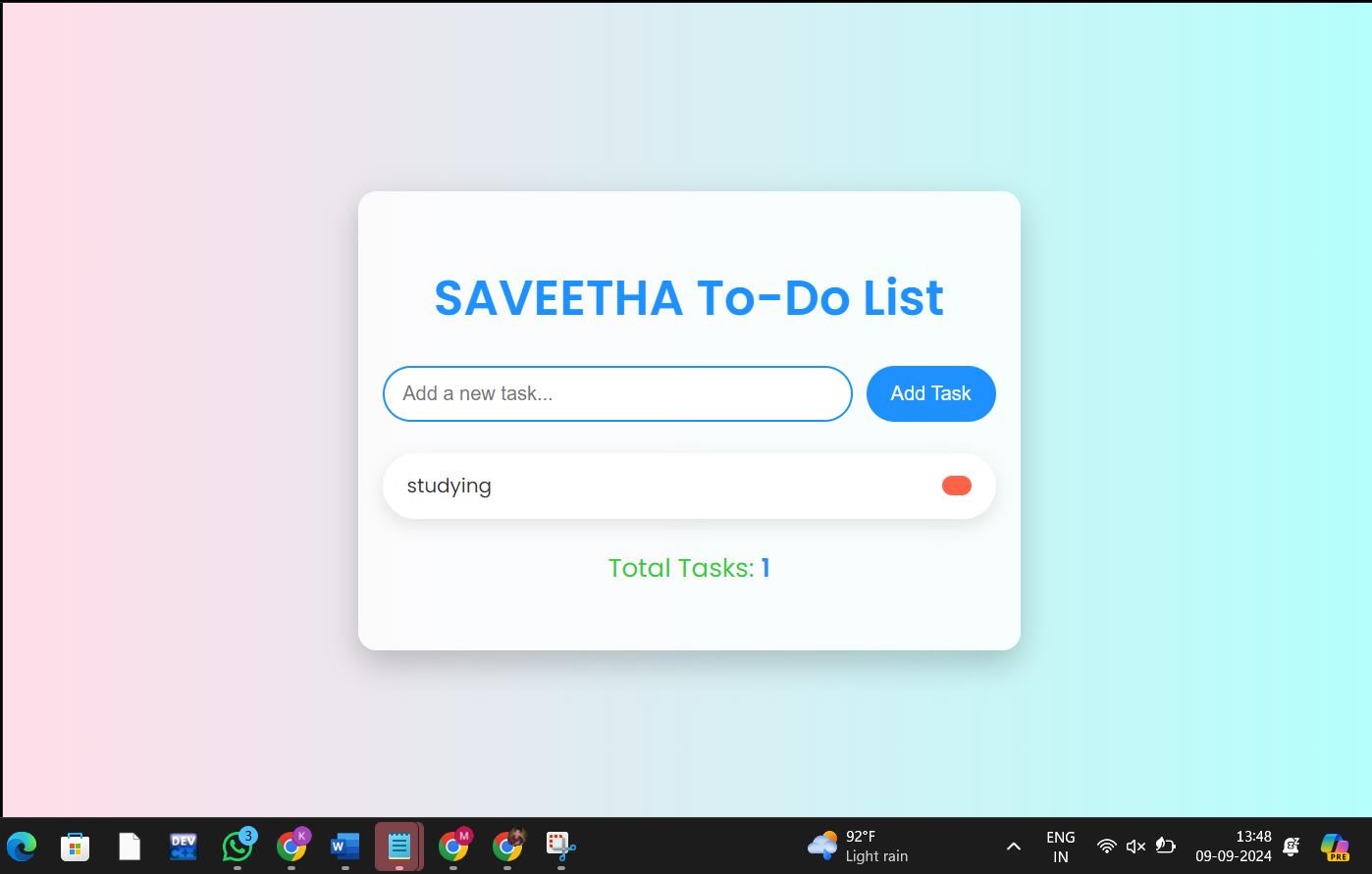


Figure 8.2 : Adding the Task

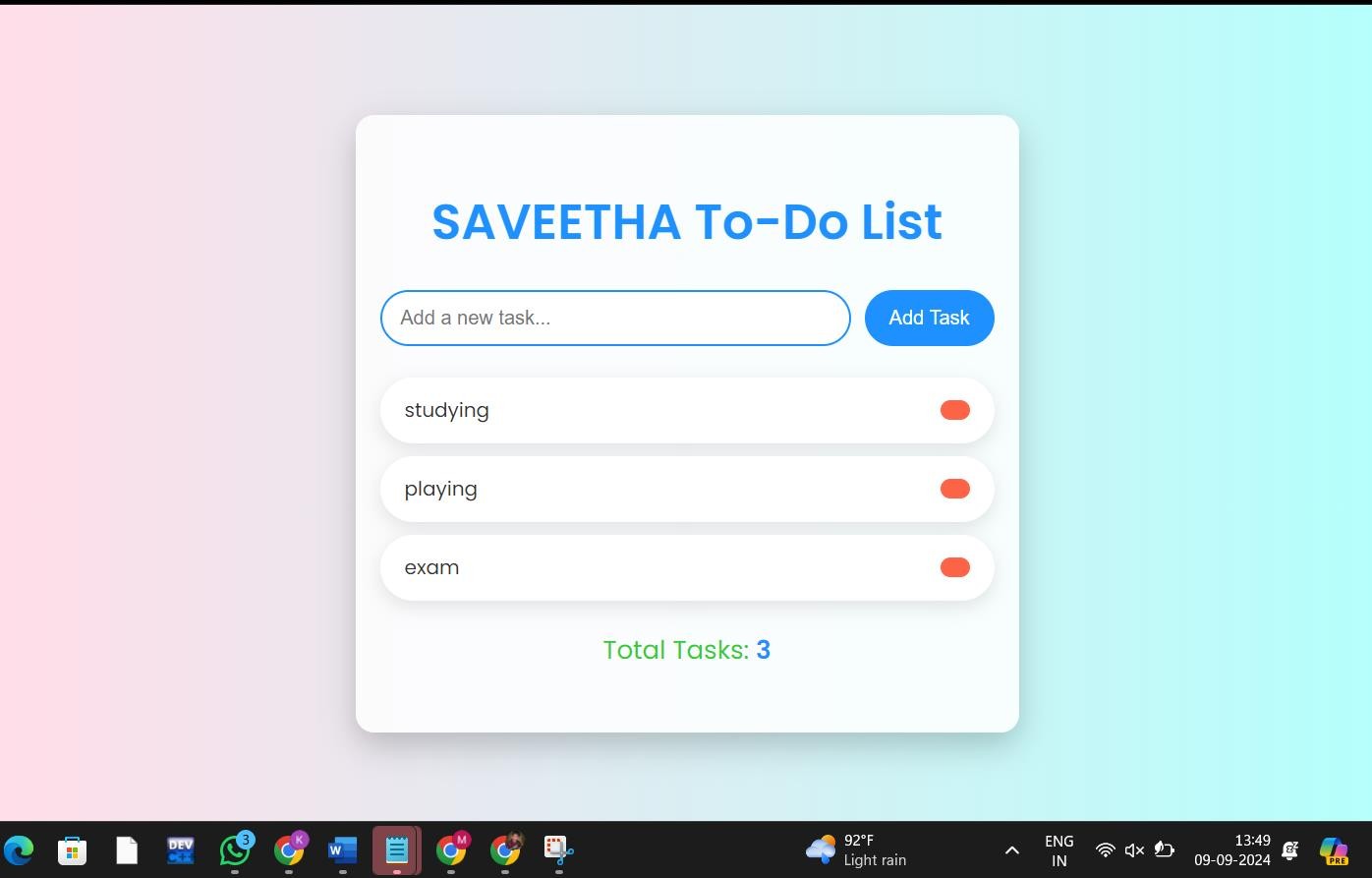


Figure 8.3 : Adding the more tasks

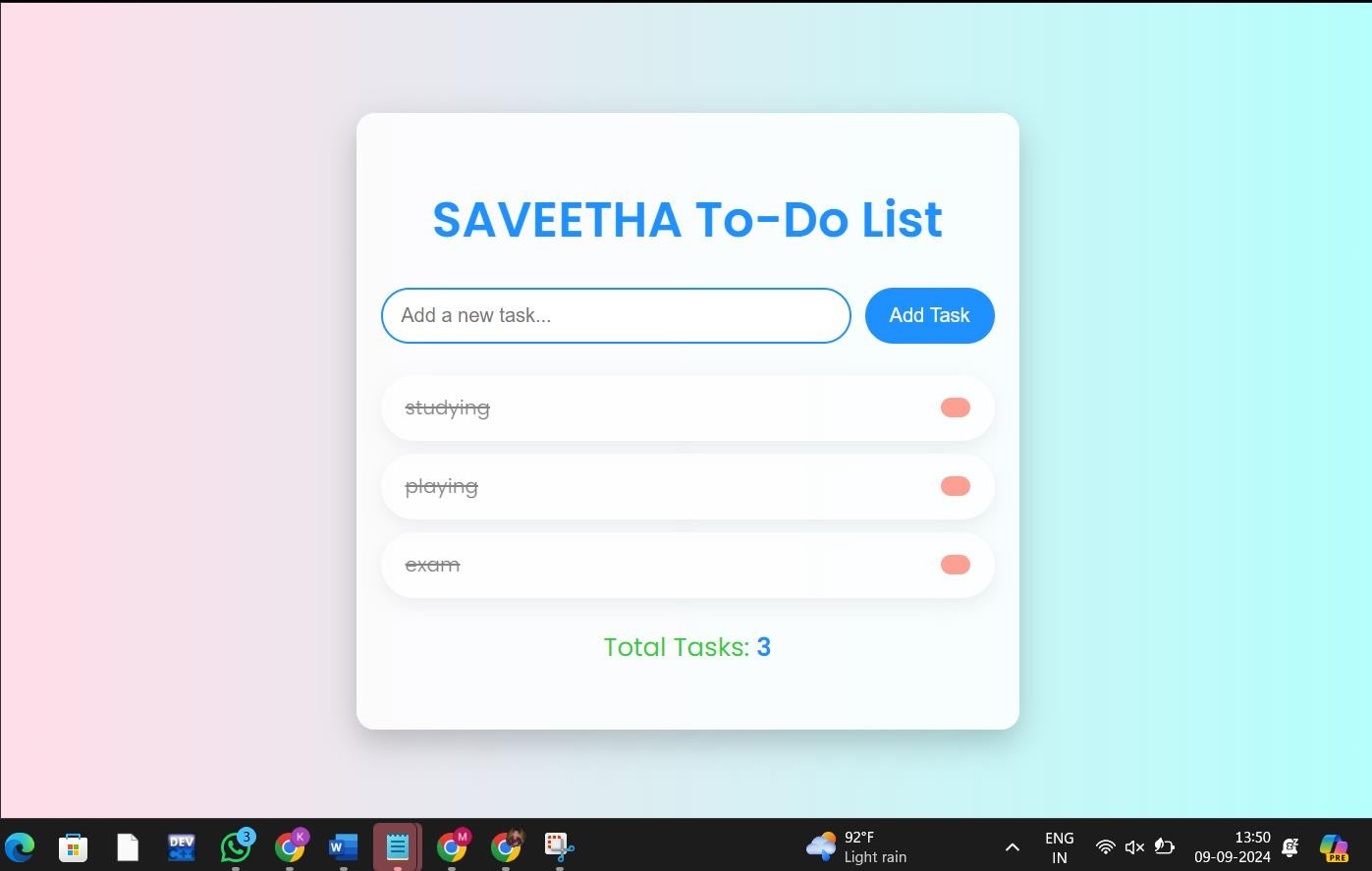


Figure 8.4 : After completing the tasks we can strike them off

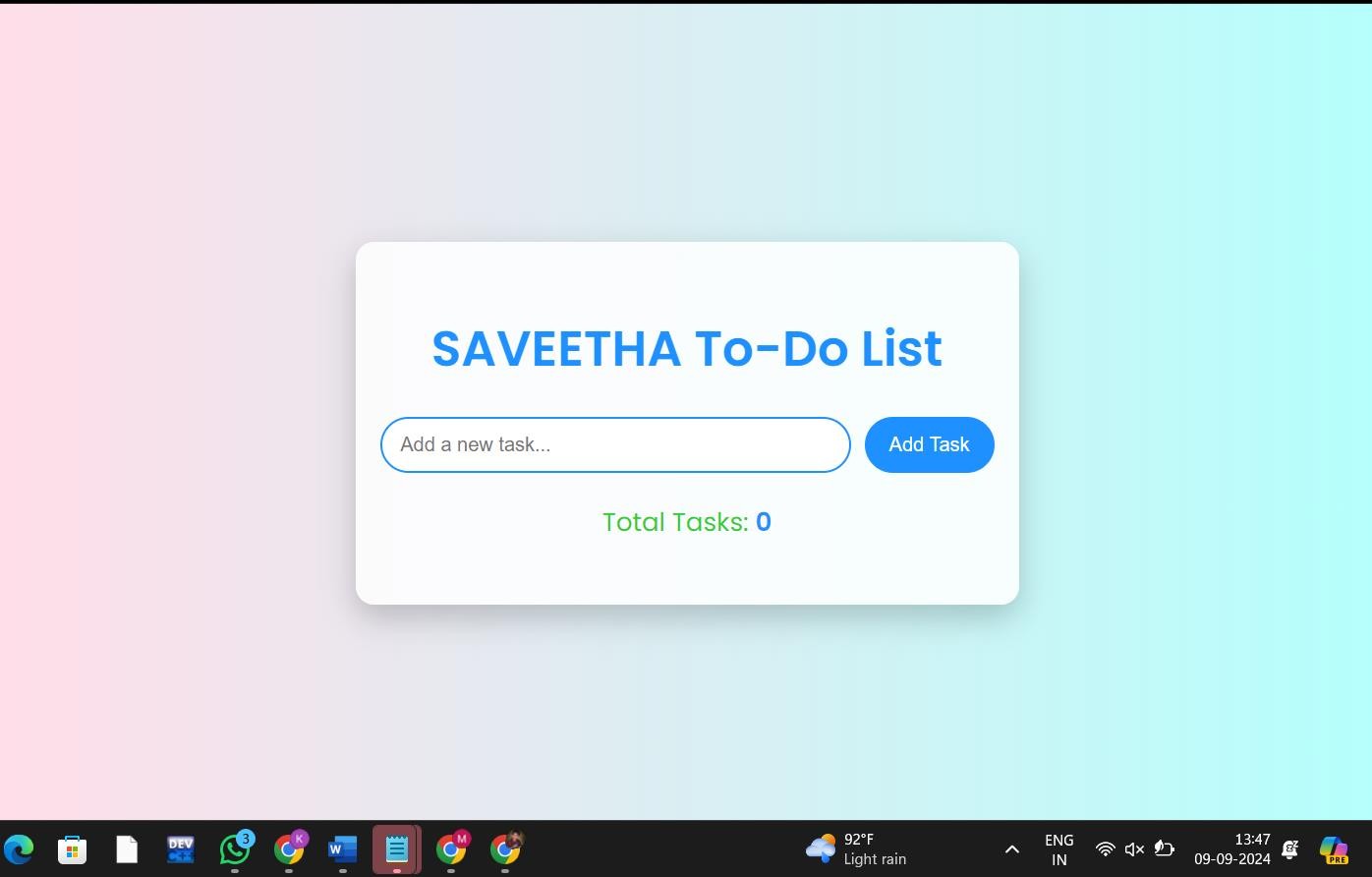


Figure 8.5 : Removed the tasks after completing it

# CONCLUSION

This capstone project successfully developed a to-do list application is designed to offer an

intuitive and efficient way to manage tasks, leveraging a well-structured architecture and clear

class interactions. By separating the application into distinct layers—Presentation, Business

Logic, and Data—this system ensures modularity, maintainability, and scalability. The class

diagram details the interactions between key components such as Task, Timer, Task Manager,

and User Interface, highlighting their roles in task management and user interaction.

The **Task** class handles individual task details and operations, while the **Timer** class manages

associated timing functionalities. The **Task Manager** class oversees the collection and

manipulation of tasks, ensuring that all task-related operations are performed smoothly.

The **User Interface** class facilitates user interactions, providing a seamless experience for task

input, display, and notifications.

# REFERENCES

* + 1. **David Allen** (2001). *Getting Things Done: The Art of Stress-Free Productivity*.

Penguin Books.

A seminal work on personal productivity and task management.

* + 1. **Cal Newport** (2016). *Deep Work: Rules for Focused Success in a Distracted World*. Grand Central Publishing.

Discusses productivity and effective task management in a digital age.

* + 1. **James Clear** (2018). *Atomic Habits: An Easy & Proven Way to Build Good Habits & Break Bad Ones*. Avery.

Focuses on habit formation and task management techniques.

* + 1. **Tim Ferriss** (2007). *The 4-Hour Workweek: Escape 9-5, Live Anywhere, and Join the New Rich*. Crown Publishing Group.

Offers strategies for time management and increasing productivity.

* + 1. **Stephen R. Covey** (1989). *The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change*. Free Press.

Includes principles of effective personal and professional task management.

* + 1. **Brian Tracy** (2015). *Eat That Frog!: 21 Great Ways to Stop Procrastinating and Get More Done in Less Time*. Berrett-Koehler Publishers.

Provides practical tips for overcoming procrastination and managing tasks.

* + 1. **S.J. Scott** (2015). *Habit Stacking: 97 Small Life Changes That Take Five Minutes or Less*. S.J. Scott.

Discusses methods for task management through small habit changes.

**Gretchen Rubin** (2015). *Better Than Before: Mastering the Habits of Our Everyday Lives*. Crown Publishing Group.

Examines habit formation and management strategies.

**Charles Duhigg** (2014). *The Power of Habit: Why We Do What We Do in Life and Business*. Random House.

Explores how habits influence productivity and task management.

**Mark Forster** (2011). *Do It Tomorrow and Other Secrets of Time Management*. Hodder & Stoughton.

Provides techniques for effective time management and task completion.

**Zack Kuhlman** (2016). *The To-Do List Formula: A Stress-Free Guide to Creating and Managing To-Do Lists*. CreateSpace Independent Publishing Platform.

Offers a guide to creating and managing effective to-do lists.

**Jeff Sutherland** (2014). *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Business.

Discusses the Scrum methodology for task and project management.

**J.D. Meier** (2013). *Getting Results the Agile Way: A Personal Results System for Work and Life*.

J.D. Meier.

Focuses on agile techniques for personal productivity and task management.

**Peter Bregman** (2011). *18 Minutes: Find Your Focus, Master Distraction, and Get the Right Things Done*. Hachette Books.

Provides strategies for maintaining focus and managing tasks effectively.

**David Allen** (2015). *Making It All Work: Winning at the Game of Work and Business of Life*. Penguin Books.

Expands on task management principles and their application in various areas of life.