

A Project Report

on

TO-DO LIST APPLICATION

Submitted in partial fulfilment of requirements for the award of the course

of

**CGA1121 – DATA STRUCTURES**

Under the guidance of

### Mrs. K. MAKANYADEVI M.E.,

### Assistant Professor/CSE

### Mrs.M.SARATHA., B.Tech., M.E.,

### Assistant Professor/AIDS

Submitted By



**M. S. ARUN SANJEEV ( 927623BCS011 )**

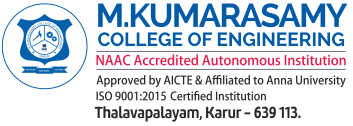
**DEPARTMENT OF FRESHMAN ENGINEERING**

**M.KUMARASAMY COLLEGE OF ENGINEERING**

(Autonomous)

**KARUR – 639 113**

MAY 2024



# M. KUMARASAMY COLLEGE OF ENGINEERING

**(Autonomous Institution affiliated to Anna University, Chennai)**

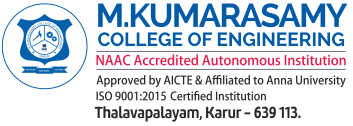
# KARUR – 639 113

**BONAFIDE CERTIFICATE**

Certified that this project report on **“ TO-DO LIST APPLICATION ”** is the bonafide work of **M S ARUN SANJEEV (927623BCS011)** who carried out the project work during the academic year 2023- 2024 under my supervision.

|  |  |
| --- | --- |
| Signature | Signature |
| **Mrs. P. KAYALVIZHI M.E.,** | **Dr. K.CHITIRAKALA, M.Sc., M.Phil.,Ph.D.,** |
| **SUPERVISOR,** | **HEAD OF THE DEPARTMENT,** |
| Department of Computer Science | Department of Freshman Engineering, |
| and Engineering, |  |
| M. Kumarasamy College of Engineering, | M. Kumarasamy College of Engineering, |
| Thalavapalayam, Karur -639 113. | Thalavapalayam, Karur -639 113. |





**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**VISION OF THE INSTITUTION**

To emerge as a leader among the top institutions in the field of technical education

**MISSION OF THE INSTITUTION**

* Produce smart technocrats with empirical knowledge who can surmount the global challenges
* Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
* Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

**VISION OF THE DEPARTMENT**

To achieve education and research excellence in Computer Science and Engineering

**MISSION OF THE DEPARTMENT**

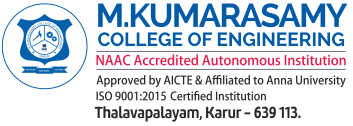
* To excel in academic through effective teaching learning techniques
* To promote research in the area of computer science and engineering with the focus on innovation
* To transform students into technically competent professionals with societal and ethical responsibilities

**PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO 1:** Graduates will have successful career in software industries and R&D divisions through continuous learning.

**PEO 2:** Graduates will provide effective solutions for real world problems in the key domain of computer science and engineering and engage in lifelong learning.

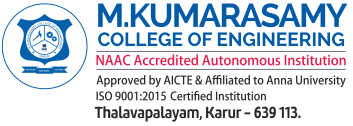
**PEO 3:** Graduates will excel in their profession by being ethically and socially responsible.



**PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

1. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
2. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
3. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

1. **PSO1: Professional Skills:** Ability to apply the knowledge of computing techniques to design and develop computerized solutions for the problems.
2. **PSO2: Successful career:** Ability to utilize the computing skills and ethical values in creating a successful career.

# ABSTRACT

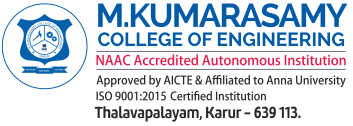
The "To-Do List Application" project, developed using the C programming language, aims to create a simple, efficient, and user-friendly solution for managing daily tasks. This application addresses common productivity challenges by allowing users to create, update, delete, and view tasks systematically. Built with robust data structures such as linked lists, hash maps, and stacks/queues, the application ensures optimal performance and usability.

The primary objective of this project is to develop a task management tool that helps users organize their work and personal tasks effectively. By implementing core functionalities like task addition, deletion, editing, and sorting, the application offers a comprehensive solution for task management. Furthermore, advanced features such as task search and filtering enhance the user experience, making it easier to locate specific tasks and manage extensive task lists.

The system design focuses on a modular architecture, with clear separation between the user interface, business logic, and data storage. This approach facilitates maintenance and scalability. The application is developed using the C programming language, utilizing libraries such as the standard I/O library for console input/output operations. A linked list is employed for maintaining task order, ensuring efficient insertion and deletion operations. A hash map provides quick access to tasks by their unique identifiers, while stack/queue structures support undo/redo functionality.

Implementation involved setting up a development environment with a C compiler and an Integrated Development Environment (IDE) such as Code Blocks or Visual Studio Code. Best coding practices were followed to ensure a robust and maintainable codebase. Each feature was meticulously coded and tested, ensuring that the application met all functional requirements. Testing involved a comprehensive suite of test cases covering all aspects of functionality, from task creation to deletion, ensuring reliability and accuracy.

In conclusion, the "To-Do List Application" project demonstrates the practical application of data structures in software development, providing a valuable tool for task management. The project not only meets its initial goals but also lays the groundwork for future improvements, highlighting the potential for further development and integration of advanced features.

# ABSTRACT WITH POs AND PSOs MAPPING

|  |  |  |
| --- | --- | --- |
| **ABSTRACT** | **POs MAPPED** | **PSOs MAPPED** |
| The "To-Do List Application" project, developed in C, aims to create an efficient and user-friendly tool for managing daily tasks. It addresses productivity challenges by allowing users to create, update, delete, and view tasks systematically. Utilizing data structures such as linked lists, hash maps, and stacks/queues, the application ensures optimal performance and usability.  The objective is to develop a task management tool for organizing tasks effectively. Core functionalities include task addition, deletion, editing, and sorting, along with advanced features like task search and filtering to enhance user experience. The system's modular design separates the user interface, business logic, and data storage, facilitating maintenance and scalability.  Development involved setting up a C environment with an IDE like VS CODE, following best practices for a robust codebase. Comprehensive testing ensured functionality and reliability. Challenges included handling large task volumes and creating an intuitive UI, addressed through data structure optimization and user-feedback. | **PO1(2)**  **PO2(3)**  **PO3(2)**  **PO4(2)**  **PO5(3)**  **PO6(1)**  **PO7(3)**  **PO8(2)**  **PO9(3)**  **PO10(3)**  **PO11(2)**  **PO12(2)** | **PSO1(3)**  **PSO2(2)** |

**Note:** 1- Low, 2-Medium, 3- High

**SUPERVISOR HEAD OF THE DEPARTMENT**

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** No. | TITLE | **PAGE** No. |
| 1 | Introduction |  |
|  | 1.1 Introduction |  |
|  | 1.2 Objective |  |
|  | 1.3 Data Structure Choice |  |
| 2 | Project Methodology |  |
|  | 2.1 |  |
|  | 2.2 Block Diagram |  |
| 3 | Modules |  |
|  | 3.1 Module 1 |  |
|  | 3.2 Module 2 |  |
|  | 3.3 Module 3 |  |
| 4 | Results and Discussion |  |
| 5 | Conclusion |  |
|  | References |  |
|  | Appendix |  |

# CHAPTER 1

# INTRODUCTION

# 1.1 Introduction

The "To-Do List Application" is a project aimed at developing an efficient and user-friendly task management tool using the C programming language. In today's fast-paced environment, managing tasks efficiently is crucial to maintaining productivity and meeting deadlines. Many people face challenges in organizing their daily activities, leading to inefficiencies and missed commitments. This application seeks to address these issues by providing a digital solution that allows users to easily create, manage, and track their tasks.

**1.2 Objective**

The primary objective of the "To-Do List Application" project is to develop a software tool that helps users organize their tasks effectively. The specific objectives are:

* **User-Friendly Interface**: Create an intuitive interface that allows users to easily add, edit, delete, and view tasks.
* **Efficient Data Management**: Implement data structures that ensure efficient storage, retrieval, and manipulation of tasks.
* **Core Functionalities**: Provide essential features such as task sorting, searching, and prioritization to enhance task management.
* **Performance Optimization**: Ensure the application performs well with a large number of tasks, maintaining quick response times.
* **Scalability**: Design the system to be easily extendable for future enhancements such as notifications and synchronization across devices.

**1.3 Data Structure Choice**

The choice of data structures is crucial for the performance and functionality of the "To-Do List Application." The following data structures were selected based on their efficiency and suitability for the tasks at hand:  
  
**Linked List:**  
  
***Purpose:*** To maintain the list of tasks in an ordered manner.  
***Advantages:*** Dynamic size: Can easily grow and shrink as tasks are added or removed.  
***Efficient insertion and deletion:*** Adding or removing tasks can be done in constant time when the position is known.  
***Usage:*** Used for the primary list of tasks, allowing efficient traversal and manipulation.  
  
**Stack/Queue:**  
  
***Purpose:*** To implement features like undo/redo functionality.  
***Advantages:*** LIFO (Last In, First Out) for stack: Perfect for undo operations where the most recent action is reversed first.  
FIFO (First In, First Out) for queue: Suitable for features like task reminders or notifications where tasks are handled in the order they were added.  
***Usage:*** Used to manage the history of user actions for undo/redo operations and for implementing task reminders.

**CHAPTER 2**

**PROJECT METHODOLOGY**

#### 2.1 Queue

In the "To-Do List Application" project, queues are a fundamental data structure utilized to manage specific functionalities efficiently. Here’s some brief information about queues used in this project:

**Purpose:**

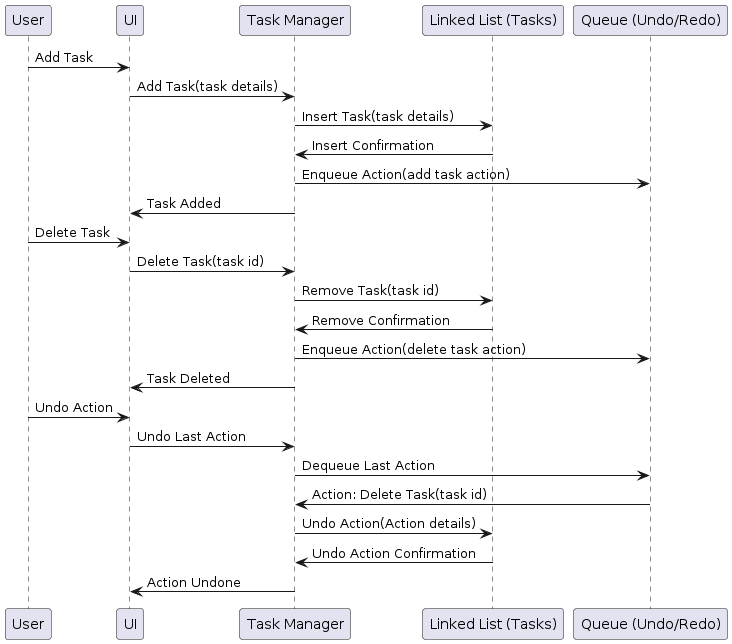
Queues are used to manage undo/redo operations and task reminders, ensuring that actions and notifications are handled in a first-in, first-out (FIFO) manner. This helps in maintaining the sequence of operations and reminders as they were added.

**Implementation:**  
 Queues are implemented using linked lists, where each node in the list represents an action or reminder. The front of the queue points to the first action or reminder to be processed, while the rear points to the last one added.

**Operations:**

* **Enqueue:** Actions or reminders are added to the rear of the queue as they are performed or set by the user.
* **Dequeue:** Actions or reminders are removed from the front of the queue as they are processed or triggered.
* **IsEmpty:** Checks if the queue is empty, indicating whether there are any pending actions or reminders.
* **Front:** Retrieves the action or reminder at the front of the queue without removing it, allowing for inspection or processing.

**2.2. Block Diagram**



**CHAPTER 3**

**MODULES**

* 1. **Add Task:**

**Description:**  
 The "Add Task" functionality allows users to add new tasks to their to-do list. When the user selects this option, they are prompted to enter details such as the task name. The system then stores this information in a data structure, associating a unique identifier with each task.  
  
**Implementation:**-  
  
**Process:** Prompt user to enter task details.

* Assign a unique identifier based on the current Task Count.
* Increment Task Count.
* Store task information in the data structure.
  1. **3 View Task:**

**Description:**  
 The "View Tasks" functionality displays all tasks in the to-do list. Upon selecting this option, the system iterates through the data structure, printing details of each task.

**Implementation:**-  
  
 **Process:**

Iterate through the data structure.

Print task ID, name, and any other relevant details for each task.

**3.3 Delete Task:**

**Description:**  
 The "Delete Task" functionality allows users to remove tasks based on their task ID. When a user selects this option, they are prompted to enter the task ID of the task they want to delete. The system then searches for the task in the data structure and removes it if found.

**Implementation:**-  
  
 **Process:** Prompt user to enter the task ID to be deleted.

* Search for the task in the data structure.
* If found, remove the task.
* Display success message; otherwise, display an error message.

**3.4 Exit**

**Description:**  
 The "Exit" option allows users to gracefully terminate the to-do list application. Upon selecting this option, the system displays a termination message and exits, returning control to the operating system.

**Implementation:**-  
  
 **Process:** Displays Exit Message.

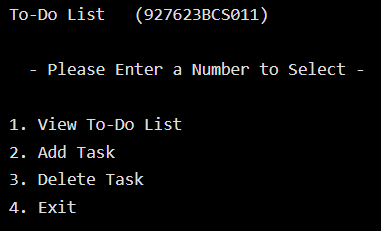
* Search for the task in the data structure.
* Terminate the Program.
* Returns to Operating System.

**CHAPTER 4**

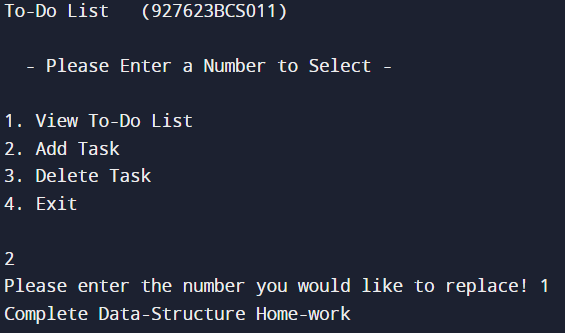
**RESULTS AND DISCUSSION**

**4.1 Results**

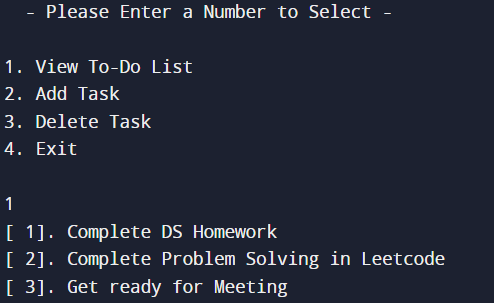
**4.1.1 User Interface Menu:**

****

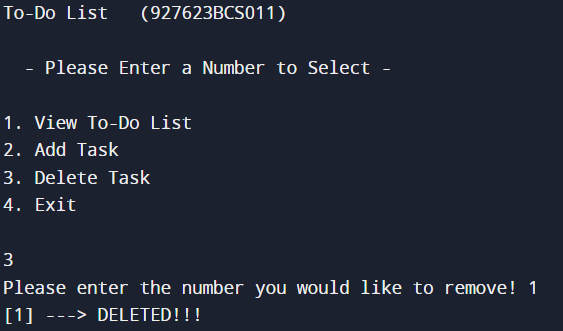
**4.1.2 Add To Do List:**

****

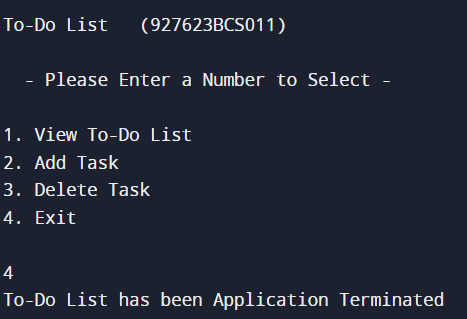
**4.1.3 View To-Do List**

****

**4.1.4 Delete Task:**

****

**4.1.5 Exit:**

****

**4.2 Discussion**

The "To-Do List Application" project aims to develop a simple yet effective tool for managing tasks efficiently. With the increasing demands on personal and professional time, individuals often struggle to organize their tasks effectively. This application addresses this challenge by providing a user-friendly interface for creating, viewing, and managing tasks.  
  
**Importance of Task Management:**Effective task management is crucial for personal productivity and goal achievement. By organizing tasks systematically, individuals can prioritize their activities, reduce stress, and improve time management skills. However, traditional methods of task management, such as pen and paper lists, are often cumbersome and lack features for efficient organization and tracking.  
  
**Purpose of the Application:**  
The primary purpose of the "To-Do List Application" is to streamline the task management process. By providing features such as task categorization, prioritization, and reminders, the application aims to enhance user productivity and organization. Additionally, the application seeks to offer a flexible and customizable platform to meet the diverse needs of users across different domains.  
  
**Discussion on Key Features:**

**Task Addition:** Users can easily add new tasks to their to-do list, specifying details such as task name and due date.  
**Task Viewing:** The application allows users to view their entire task list, providing an overview of pending tasks and their respective details.  
**Task Deletion:** Users can delete tasks from the list, removing completed or unnecessary items to declutter their workspace.  
**User-Friendly Interface:** The application features an intuitive user interface, ensuring ease of use and accessibility for users of all levels of technical proficiency.  
**Customization Options:** Users can customize their task list, organizing tasks into categories or assigning priority levels to facilitate efficient task management.  
**Reminder Functionality:** The application includes reminder features, notifying users of upcoming deadlines or important tasks to ensure timely completion.  
**Implementation Approach:**  
The application is developed using the C programming language, leveraging the principles of modular design and efficient data structures. Modular design ensures separation of concerns, allowing for easier maintenance and scalability, while efficient data structures such as linked lists and arrays optimize task storage and retrieval.

**CHAPTER 5**

**CONCLUSION**

In conclusion, the "To-Do List Application" stands as a robust solution for modern task management challenges, offering users a streamlined and intuitive platform to organize and prioritize their activities effectively. Through its user-friendly interface and customizable features, the application caters to diverse user needs, empowering individuals to take control of their tasks and optimize their productivity. By leveraging modular design principles and efficient data structures, the application ensures scalability, maintainability, and optimal performance. As we navigate the evolving landscape of task management, the "To-Do List Application" remains poised for further enhancements and advancements, promising to continue serving as an invaluable tool for individuals striving to achieve their goals efficiently in today's dynamic world.

**REFERENCES**

**1. Trello. (n.d.). Retrieved from *https://trello.com/***  
  
 Trello is a popular project management tool that offers to-do list functionality among its features. Exploring Trello's interface and user experience can provide valuable insights into effective design and functionality for to-do list applications.

***2*.Todoist. (n.d.). Retrieved from *https://todoist.com/***  
 Todoist is another widely used to-do list application that offers a range of features for task management and organization. Analyzing Todoist's features and user experience can offer inspiration and ideas for designing and implementing similar functionalities in your own to-do list application.

**APPENDIX**

#**include** <stdio.h>  
#**include** <stdlib.h>  
#**include** <string.h>  
  
  
#**define** MAX\_STRING\_LENGTH 101  
#**define** MAX\_TODO 20  
  
  
**int** **main**()  
{  
 **int** input;  
 **int** loop;  
 FILE \*f;  
  
 **char** list[MAX\_TODO][MAX\_STRING\_LENGTH] = {};  
@requires\_authorization  
def somefunc(param1='', param2=0):  
 r'''A docstring'''  
 if param1 > param2: # interesting  
 print 'Gre\'ater'  
 return (param2 - param1 + 1 + 0b10l) or None  
  
class SomeClass:  
 pass  
  
>>> message = '''interpreter  
... prompt'''

**while**(1) {  
  
 printf("\nTo-Do List (927623BCS011) \n\n");  
 printf(" - Please Enter a Number to Select - \n\n");  
 printf("1. View To-Do List \n");  
 printf("2. Add Task \n");  
 printf("3. Delete Task \n");  
 printf("4. Exit\n\n");  
  
 scanf("%d", &input);  
  
  
 **if**(input == 4) {  
 f = fopen("list.bin", "w");  
 fwrite(list, **sizeof**(**char**), **sizeof**(list), f);  
 fclose(f);  
 **return** EXIT\_SUCCESS;  
 } **else** **if** (input == 1 || input == 2 || input == 3) {  
 **switch**(input) {  
 **case** 1:  
 **for** (**int** loop = 0; loop < MAX\_TODO; loop++) {  
 printf("[%2d]. %s\n", loop + 1, list[loop]);  
 }  
 **break**;  
 **case** 2:  
 printf("Please enter the number you would like to replace! ");  
 **int** j;  
 scanf("%d", &j);  
  
 getchar();  
 scanf("%[^\n]", list[j - 1]);  
 **break**;  
 **case** 3:  
 printf("Please enter the number you would like to remove! ");  
 **int** k;  
 scanf("%d", &k);  
  
 strncpy(list[k-1], " ", 101);  
  
 printf("[%d] ---> DELETED!!!", k);  
 **break**;  
 }  
 } **else** {  
 printf("Please enter 1, 2, 3, or 4...\n");  
 }  
  
 }  
  
 **return** EXIT\_SUCCESS;  
}