TO-DO APPLICATION

IN PYTHON

A PROJECT REPORT

Submitted by

THARUN KUMAR M (2303811710621114)

in partial fulfillment for the completion of the course

ECA1121- PYTHON PROGRAMMING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE, 2024

K.RAMAKRISHNAN
COLLEGE OF TECHNOLOGY
An Autonomous Institution
Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015 & ISO 14001:2015 Certified Institution, Accredited with 'A+' grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.
Incubating Minds
Catalyzing Careers

BONAFIDE CERTIFICATE

Certified that this project report titled **"TO-DO APPLICATION"** is the bonafide work of **THARUN KUMAR M (2303811710621114)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a course was conferred on an earlier occasion on this or any other candidate.

15/6/24

15/6/24

SIGNATURE

Dr.S.SYEDAKBAR, M.E.Ph.D

HEAD OF DEPARTMENT

ASSISTANT PROFESSOR

Department of Electronics and Communication Engineering

K Ramakrishnan College of Technology (Autonomous), Samayapuram- 621 112

SIGNATURE

Mrs.P.SUDHA, M.E., (Ph.D)

SUPERVISOR

ASSISTANT PROFESSOR

Department of Electronics and Communication Engineering

K Ramakrishnan College of Technology (Autonomous), Samaypuram- 621 112

Submitted for the viva-voce examination held on 15.06.2024

DECLARATION

I declare that the project report on **"TO-DO APPLICATION IN PYTHON"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **ECA1121- PYTHON PROGRAMMING**.

Signature

THARUN KUMAR M

Place: Samayapuram

Date: 15.06.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. SYEDAKBAR, M.E.,Ph.D.,** Head of the department, **ELECTRONICS AND COMMUNICATION ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project guide **Mrs., k. KARPOORA SUNDARI, M.E.,(P.D)** Department of **ELECTRONICS AND COMMUNICATION ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

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 DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

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

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

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

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

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

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

Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

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.

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.

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.

ABSTRACT

“To-do application in Python" is a Python script designed to development of a versatile task management application designed to cater to the diverse needs of users ranging from students and professionals to homemakers. The application seeks to address the shortcomings of existing task management solutions, which often fall into two categories: overly simplistic to-do list applications and complex project management tools. Simplistic applications lack essential features such as due date management, task prioritization, and progress tracking, making them inadequate for users with more sophisticated needs. On the other hand, complex project management tools, while feature-rich, are often daunting and overwhelming for users looking for straightforward task management, leading to a steep learning curve and underutilization of their capabilities. It strike a balance by developing an application that is both feature-rich and user-friendly. The application allows users to seamlessly add, edit, delete, and mark tasks as completed, as well as set due dates to keep track of deadlines. By integrating these functionalities into an intuitive and accessible interface, the application ensures that users can manage their tasks efficiently without being bogged down by unnecessary complexity. to enhance productivity by providing a centralized platform for task management that is easy to use yet powerful enough to meet the demands of users. By offering essential features in a streamlined manner, the application aims to improve users' ability to organize, prioritize, and complete their tasks, ultimately leading to better time management and increased productivity. This project represents a significant step forward in providing a practical solution to the everyday challenges of task management, combining simplicity with functionality to deliver a tool that meets the needs of its users effectively.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

IDE	-	Integrated Development Environment
OOP	-	Object Oriented Programming
CLI	-	Command Line Interrface
GUI	-	Graphical User Interface
APIs	-	Application Programming Interface

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO PYTHON

1.1.1. Overview

Python is a widely-used, high-level programming language renowned for its readability and simplicity, making it an ideal choice for both novice and seasoned programmers. Created by Guido van Rossum and released in 1991, Python's core philosophy emphasizes code readability and straightforward syntax, allowing developers to write clear and concise code more efficiently compared to other languages like C++ or Java.

1.1.2. Programming Paradigms

Python supports various programming paradigms, including procedural, object-oriented, and functional programming. This flexibility, combined with a dynamic type system and automatic memory management, facilitates the development of a wide range of applications, from simple scripts to complex software systems.

1.1.3. Standard Library

The language's comprehensive standard library, often referred to as "batteries-included," provides built-in modules and functions for handling many programming tasks, such as file I/O, system calls, and even web services. This extensive library helps streamline the development process by offering ready-to-use solutions for common programming challenges.

1.1.4. Third-Party Libraries And Frameworks

One of Python's significant strengths is its extensive ecosystem of third-party libraries and frameworks. Popular libraries such as NumPy and Pandas enable efficient data manipulation and analysis, while frameworks like Django and Flask streamline web development. In the realm of machine learning and artificial intelligence, libraries like TensorFlow and PyTorch are widely adopted for building and deploying sophisticated models.

1.1.5. Versions Of Python

Python has undergone significant evolution since its inception, with two major versions in use today:

Python 2: Released in 2000, Python 2.x series was a major milestone and widely used for many years. However, it reached its end of life on January 1, 2020, and is no longer maintained.

Python 3: Introduced in 2008, Python 3.x series brought substantial improvements and changes to the language, such as better Unicode support, a more consistent syntax, and enhanced standard libraries. Python 3 is the recommended version for all new projects.

1.1.6. Python Tools

Python's ecosystem includes numerous tools that enhance productivity and development experience:

IDEs and Code Editors: Popular options include PyCharm, VS Code, and Jupyter Notebook, which offer features like syntax highlighting, code completion, and debugging.

Package Management: Tools like pip and conda facilitate the installation and management of Python libraries and dependencies.

Virtual Environments: virtualenv and venv allow developers to create isolated environments for different projects, ensuring dependency conflicts are avoided.

Testing Frameworks: unittest, pytest, and nose are commonly used for writing and running tests to ensure code reliability and correctness.

Build Tools: setuptools and wheel help in packaging Python projects, making them easy to distribute and install.

Documentation Generators: Tools like Sphinx are used to create comprehensive documentation for Python projects.

Linters and Formatters: pylint, flake8, and black help maintain code quality and consistency by enforcing coding standards and formatting.

1.1.7. Versatility And Adoption

Python's simplicity and versatility have led to its widespread adoption in various fields, including web development, data science, artificial intelligence, automation, and scientific computing. Its active community continually contributes to a rich repository of resources, tutorials, and documentation, making it easier for developers to learn and apply Python effectively.

CHAPTER 2

PROJECT DESCRIPTION

2.1. PROJECT INTRODUCTION

The project, "To-Do Application in Python," aims to involves the creation of a task management application designed to help users efficiently organize and track their daily activities. The application is intended to be both powerful and user-friendly, providing essential features that make task management straightforward and effective. By focusing on simplicity and functionality, the application aims to meet the needs of users from various backgrounds, whether they are students, professionals, or homemakers. This tool will serve as a centralized platform where users can manage their tasks, set deadlines, and monitor their progress in an intuitive manner. The ultimate goal is to enhance productivity and ensure that users can stay on top of their responsibilities without feeling overwhelmed.

2.2. PROJECT OBJECTIVE

The primary objective of this project is to develop a robust and intuitive task management application that simplifies the process of organizing and tracking tasks. The application aims to provide users with essential functionalities such as adding, editing, deleting tasks, setting due dates, and marking tasks as completed. By offering these features in a user-friendly interface, the project seeks to improve task management efficiency and productivity.

2.3. PROBLEM STATEMENT

Effective task management is a common challenge in today's fast-paced environment, where individuals and professionals often juggle multiple responsibilities. Existing solutions are either too basic, lacking crucial features like due dates and task prioritization, or too complex, with a steep learning curve and unnecessary functionalities. This gap in the market means that users struggle to find a tool that balances simplicity with functionality. The problem is further compounded by the need for a centralized platform that can efficiently organize, prioritize, and track tasks. This project aims to address these issues by developing a task management application that combines ease of use with essential features, providing an effective solution for managing tasks and deadlines.

2.4. LIBRARIES USED

The development of this task management application leverages several key Python libraries to ensure robust functionality and ease of use. The datetime library is utilized to handle date and time operations, allowing users to set and manage task due dates accurately. The sys library is employed to interact with the Python interpreter, enabling smooth program execution and graceful exit operations when necessary. Additionally, the os library is used for file and directory operations, supporting the application in tasks such as saving and retrieving user data. These libraries are chosen for their reliability and efficiency, ensuring that the application performs well and meets user expectations for a seamless task management experience.

CHAPTER 3

SYSTEM ANALYSIS

3.1. EXISTING SYSTEM

Existing task management systems range from traditional paper-based planners to digital to-do list applications and comprehensive project management tools. Paper-based planners, while simple and tangible, lack digital capabilities such as reminders and automatic updates. Basic to-do list apps provide a digital alternative but often miss critical features like due date management and task prioritization.. These existing systems, therefore, fall short in balancing simplicity with essential features, making them inadequate for effective task management

3.1.1. DISADVANTAGES

a. Paper-based Planners

Prone to loss or damage, require manual updates, lack digital features like reminders, and are not easily accessible remotely.

b. Basic T0-do apps

Lack advanced features like due dates and prioritization, have limited task tracking, can't handle complex projects, and lack integration with other tools.

c. Complex Project Management Tools

These tools can be overwhelming due to their extensive features, which are often unnecessary for simple tasks. They require significant time to learn, can be expensive, and may be underutilized. Security concerns and dependence on internet connectivity are additional disadvantages.

d. Resource Intensive

These tools can be resource-intensive, requiring significant system resources and potentially slowing down other applications.

e. Scalability Issues

For very large projects, even advanced tools may struggle with performance and data management.

f. Team Coordination

Complexity can make it challenging for all team members to be on the same page, especially if they are not equally proficient with the tool.

3.2. PROPOSED SYSTEM

The proposed system, "To-Do application in Python," aims to fill the gap between overly simplistic and overly complex task management solutions by offering a balanced application that is both feature-rich and user-friendly. By focusing on simplicity and accessibility, the proposed system ensures that users can manage their tasks efficiently without feeling overwhelmed by unnecessary features. The application is designed to cater to users with varying levels of technical expertise, making it a versatile tool for enhancing productivity across different user demographics.

3.2.1. ADVANTAGES

a. User-Friendly Interface

The proposed system is designed with a user-friendly interface that is intuitive and easy to navigate. This design reduces the learning curve for new users, allowing them to quickly become proficient in using the application. The interface enables users to efficiently add, edit, delete, and manage tasks, enhancing their overall experience and productivity.

b. Essential Features

The proposed system provides critical functionalities such as due date setting, task prioritization, and completion tracking. These features offer a comprehensive solution for all task management needs. By balancing simplicity with functionality, the application avoids unnecessary complexity, ensuring that users can focus on their tasks without being overwhelmed by extraneous features.

c. Efficiency and Productivity

The proposed system enhances productivity by helping users stay organized and on top of their tasks. It reduces the risk of missed deadlines and forgotten responsibilities by providing a streamlined process for task management. This efficiency saves time and effort, enabling users to allocate their resources more effectively and achieve their goals.

d. Accessibility

The proposed system is designed to be accessible to users with varying levels of technical expertise, ensuring broad usability. It caters to different demographics, making it suitable for a wide range of users. The application is available across multiple devices, allowing for remote access and convenience, ensuring that users can manage their tasks from anywhere at any time.

e. Integration and Customization

The proposed system offers the potential for integration with other productivity tools and applications, enhancing its utility and adaptability. It provides customization options that allow users to tailor the application to their individual needs and preferences.

CHAPTER 4

SYSTEM DESIGN & MODULES

4.1. BLOCK DIAGRAM

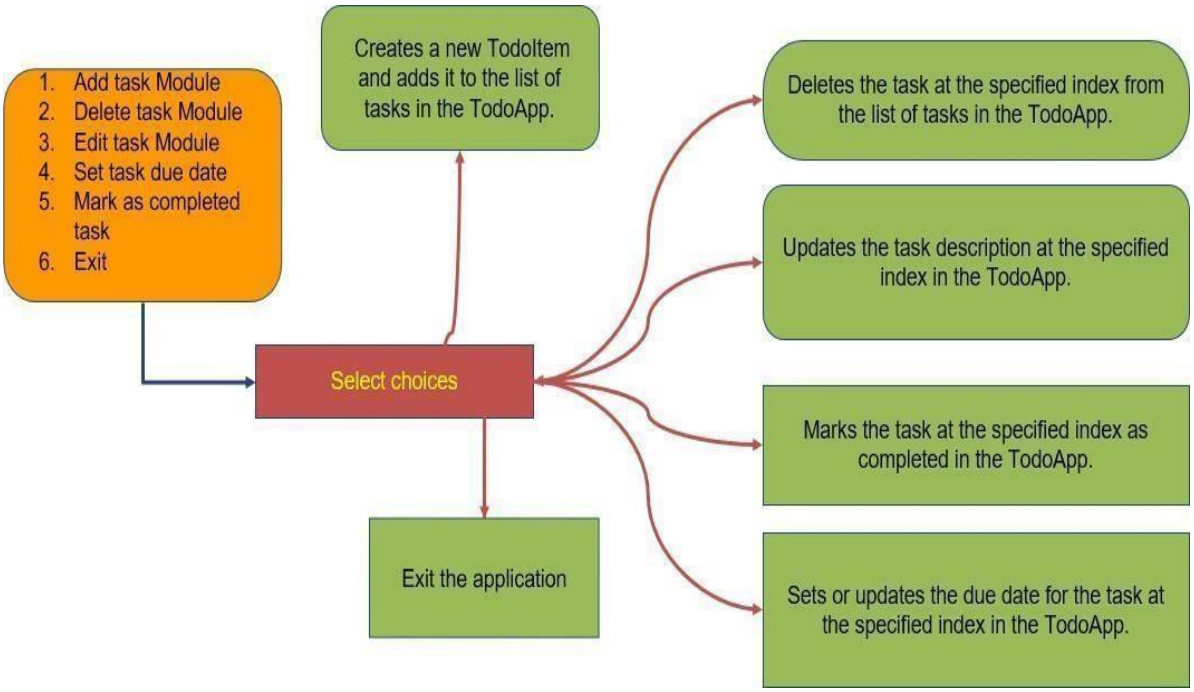

Fig. 4.1. Block Diagram

4.2. MODULE DESCRIPTION

4.2.1. USER INTERFACE MODULE

Description: Handles the presentation and interaction layer of the To-Do program, providing a command-line interface (CLI) for users to manage tasks effectively.

Functionalities:

Display menu options (display_menu() function).

Receive user input and execute corresponding actions based on selected options.

4.2.2. TASK MANAGEMENT MODULE

This module Handles core operations like adding, editing, and deleting tasks. Provides a user-friendly interface for efficient task management.

Features:

Add: Allows users to add new tasks with descriptions.

Edit: Enables users to modify existing task details.

Delete: Provides functionality to remove tasks that are no longer needed.

View/List: Displays tasks based on various criteria such as due date or priority.

4.2.3. DUE DATE MANAGEMENT MODULE

Manages task deadlines by allowing users to set, update, and receive notifications for due dates. Helps users prioritize tasks effectively.

Features:

Set Due Dates: Allows users to assign deadlines to tasks.

Notifications: Sends reminders and notifications for approaching due dates.

View by Due Date: Organizes tasks according to their deadlines.

Snooze/Dismiss: Provides options to delay or dismiss reminders.

4.2.4. DATA STORAGE MODULE

Description:Handles the persistence of task data, ensuring tasks are stored securely and retrieved reliably across sessions.

Functionalities:

Store tasks in memory (tasks list) during program execution.

Provide functions to manipulate task data (add, delete, edit, mark completed).

Potential extension: Implement file or database storage for persistent data across program runs.

CHAPTER 5

CONCLUSION & FUTURE ENHANCEMENT

5.1. CONCLUSION

The "To- Do Application in Python" project presents an effective solution for handling Tasks and due time.the To-Do program developed provides a robust foundation for effective task management through a command-line interface (CLI). Throughout its implementation, key modules were established to handle task operations, user interface interaction, data storage, and potential integration with notifications. The program successfully enables users to add, edit, delete, and view tasks, ensuring flexibility and ease of use in organizing daily activities.

The User Interface module serves as the gateway for user interaction, presenting a clear menu structure and guiding users through task management functionalities. This module ensures intuitive navigation and seamless execution of actions, enhancing the overall user experience.

The Task Management module plays a pivotal role in the program by facilitating core operations such as task creation, modification, and deletion. It ensures tasks are organized efficiently and can be easily managed according to user preferences. The implementation allows for flexible task handling, including marking tasks as completed and setting due dates when needed.

Furthermore, the Data Storage module ensures data integrity and persistence across sessions, albeit within the current scope using in-memory storage. Future enhancements could extend this module to incorporate file-based or database storage, enabling seamless data management and persistence beyond program execution.

While the current implementation does not include a Notifications module due to CLI limitations, future enhancements could integrate this feature. Notifications would provide timely reminders for tasks nearing their due dates, enhancing productivity and ensuring users stay informed about their task deadlines.

5.2. FUTURE ENHANCEMENT

Graphical User Interface (GUI): Transitioning from CLI to a GUI would significantly improve user interaction and visual appeal. A GUI could offer a more intuitive interface with drag-and-drop functionalities, task grouping, and calendar views for better task organization.

Database Integration: Implementing a database backend (e.g., SQLite, MySQL) for data storage would enhance scalability and data management capabilities. It would enable

persistent storage of tasks across sessions and support advanced querying and reporting features.

Notification System: Introducing a Notification module to send reminders and alerts for upcoming task deadlines. Integration with system-level notifications or external services (e.g., email, SMS) would keep users informed and improve task management efficiency.

Task Prioritization and Filtering: Enhancing task management capabilities with features for prioritizing tasks (e.g., high, medium, low) and filtering tasks based on criteria such as due date, priority, or category.

Collaboration Features: Adding collaborative features to allow multiple users to share and manage tasks within the same interface. This could include task assignment, comments, and real-time updates.

Reporting and Analytics: Implementing reporting tools to generate insights into task completion rates, productivity trends, and user performance metrics. Visualizations such as charts and graphs would help users analyze their task management habits and identify areas for improvement.

Customization Options: Providing users with customizable settings for task views, themes, and notification preferences to tailor the application to individual preferences and workflows.

Accessibility Improvements: Ensuring the application is accessible to users with disabilities by adhering to accessibility standards and providing keyboard shortcuts, screen reader support, and contrast adjustments.

APPENDICES

APPENDIX A-SOURCE CODE

```
import datetime

class TodoItem:
    def __init__(self, description, due_date=None):
        self.description = description
        self.completed = False
        self.due_date = due_date

    def __str__(self):
        status = "Completed" if self.completed else "Pending"
        due_date_str = self.due_date.strftime("%Y-%m-%d") if self.due_date else "No
due date"
        return f"{self.description} - {status} - Due: {due_date_str}"

class TodoApp:
    def __init__(self):
        self.tasks = []

    def add_task(self, description, due_date=None):
        task = TodoItem(description, due_date)
        self.tasks.append(task)
        print("Task added!")

    def delete_task(self, index):
        try:
            self.tasks.pop(index)
            print("Task deleted!")
        except IndexError:
            print("Invalid index!")

    def edit_task(self, index, new_description):
        try:
            self.tasks[index].description = new_description
            print("Task edited!")
        except IndexError:
            print("Invalid index!")

    def mark_as_completed(self, index):
        try:
            self.tasks[index].completed = True
            print("Task marked as completed!")
        except IndexError:
            print("Invalid index!")

    def set_due_date(self, index, due_date):
        try:
            self.tasks[index].due_date = due_date
            print("Due date set!")
        except IndexError:
            print("Invalid index!")

    def show_tasks(self):
        if not self.tasks:
            print("No tasks to show.")
        for i, task in enumerate(self.tasks):
```

```
def parse_date(date_str):
    return datetime.datetime.strptime(date_str, "%Y-%m-%d")

def main():
    app = TodoApp()

    while True:
        print("\nMenu:")
        print("1. Add task")
        print("2. Delete task")
        print("3. Edit task")
        print("4. Mark task as completed")
        print("5. Set task due date")
        print("6. Show tasks")
        print("7. Quit")

        option = input("Choose an option (1-7): ").strip()

        if option == "1":
            description = input("Task description: ").strip()
            due_date_str = input("Due date (YYYY-MM-DD) [optional]: ").strip()
            due_date = parse_date(due_date_str) if due_date_str else None
            app.add_task(description, due_date)

        elif option == "2":
            index = int(input("Task index to delete: ").strip())
            app.delete_task(index)

        elif option == "3":
            index = int(input("Task index to edit: ").strip())
            new_description = input("New task description: ").strip()
            app.edit_task(index, new_description)

        elif option == "4":
            index = int(input("Task index to mark as completed: ").strip())
            app.mark_as_completed(index)

        elif option == "5":
            index = int(input("Task index to set due date: ").strip())
            due_date_str = input("New due date (YYYY-MM-DD): ").strip()
            due_date = parse_date(due_date_str)
            app.set_due_date(index, due_date)

        elif option == "6":
            app.show_tasks()

        elif option == "7":
            break

        else:
            print("Invalid option! Please choose a number between 1 and 7.")

if __name__ == "__main__":
    main()
```

APPENDIX B -SCREEN SHOTS

1. Add task

```
Menu:
1. Add task
2. Delete task
3. Edit task
4. Mark task as completed
5. Set task due date
6. Show tasks
7. Quit
Choose an option (1-7): 1
Task description: do python programmes on codetantra
Due date (YYYY-MM-DD) [optional]: 2024-06-13
Task added!
```

2. Delete task

```
Menu:
1. Add task
2. Delete task
3. Edit task
4. Mark task as completed
5. Set task due date
6. Show tasks
7. Quit
Choose an option (1-7): 2
Task index to delete: 0
Task deleted!
```

3. Edit Task

```
Menu:
1. Add task
2. Delete task
3. Edit task
4. Mark task as completed
5. Set task due date
6. Show tasks
7. Quit
Choose an option (1-7): 3
Task index to edit: 0
New task description: do only 1st module programmes on ct
Task edited!
```

4. Mark task as completed

```
Menu:
1. Add task
2. Delete task
3. Edit task
4. Mark task as completed
5. Set task due date
6. Show tasks
7. Quit
Choose an option (1-7): 4
Task index to mark as completed: 0
Task marked as completed!
```

5. Set task due date

```
Menu:
1. Add task
2. Delete task
3. Edit task
4. Mark task as completed
5. Set task due date
6. Show tasks
7. Quit
Choose an option (1-7): 5
Task index to set due date: 0
New due date (YYYY-MM-DD): 2024-06-15
Due date set!
```

6. Show tasks

```
Menu:
1. Add task
2. Delete task
3. Edit task
4. Mark task as completed
5. Set task due date
6. Show tasks
7. Quit
Choose an option (1-7): 6
0: do only 1st module programmes on ct - Completed - Due: 2024-06-15
1: do maths record - Pending - Due: 2024-06-14
```