Description: Develop a command-line to-do list application that allows users to manage tasks.

**Features:**

1. Task Management: Allow users to add, remove, and mark tasks as completed.

2. Task Priority: Implement a priority system for tasks (e.g., high, medium, low).

3. Due Dates: Enable users to set due dates for tasks.

4. List View: Display tasks in a list with their details.

5. Data Persistence: Store tasks in a file/database for persistence across sessions.

**Tech Stack:**

• Python

• File handling or a simple database library

Creating a command-line to-do list application in Python with the specified features can be approached in a structured manner. Here's a step-by-step guide on how to develop this application:

**Step-by-Step Implementation**

1. \*Setup Project Structure\*

Create a new directory for your project and set up the initial structure:

sh

mkdir todo\_app

cd todo\_app

touch todo.py tasks.json

2. \*Task Management Functions\*

Implement functions to add, remove, and mark tasks as completed.

3. \*Task Priority and Due Dates\*

Add support for setting task priority and due dates.

4. \*List View\*

Implement a function to display the tasks in a list.

5. \*Data Persistence\*

Use a JSON file to store tasks for persistence.

**Full Implementation in todo.py**

python

import json

import os

from datetime import datetime

TASKS\_FILE = 'tasks.json'

# Load tasks from file

def load\_tasks():

if os.path.exists(TASKS\_FILE):

with open(TASKS\_FILE, 'r') as file:

return json.load(file)

return []

# Save tasks to file

def save\_tasks(tasks):

with open(TASKS\_FILE, 'w') as file:

json.dump(tasks, file, indent=4)

# Add a new task

def add\_task(tasks, title, priority, due\_date):

task = {

'title': title,

'priority': priority,

'due\_date': due\_date,

'completed': False

}

tasks.append(task)

save\_tasks(tasks)

# Remove a task by index

def remove\_task(tasks, index):

if 0 <= index < len(tasks):

tasks.pop(index)

save\_tasks(tasks)

# Mark a task as completed by index

def complete\_task(tasks, index):

if 0 <= index < len(tasks):

tasks[index]['completed'] = True

save\_tasks(tasks)

# Display the list of tasks

def list\_tasks(tasks):

if not tasks:

print("No tasks available.")

return

for idx, task in enumerate(tasks):

status = '✔' if task['completed'] else '✘'

print(f"{idx + 1}. [{status}] {task['title']} (Priority: {task['priority']}, Due: {task['due\_date']})")

**# Main command-line interface**

def main():

tasks = load\_tasks()

while True:

print("\nTo-Do List Application")

print("1. Add task")

print("2. Remove task")

print("3. Complete task")

print("4. List tasks")

print("5. Exit")

choice = input("Enter your choice: ")

if choice == '1':

title = input("Task title: ")

priority = input("Task priority (high, medium, low): ")

due\_date = input("Due date (YYYY-MM-DD): ")

add\_task(tasks, title, priority, due\_date)

elif choice == '2':

index = int(input("Task index to remove: ")) - 1

remove\_task(tasks, index)

elif choice == '3':

index = int(input("Task index to complete: ")) - 1

complete\_task(tasks, index)

elif choice == '4':

list\_tasks(tasks)

elif choice == '5':

break

else:

print("Invalid choice. Please try again.")

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Explanation of Features**

1. \*Task Management\*: Functions to add, remove, and complete tasks (add\_task, remove\_task, complete\_task).

2. \*Task Priority\*: Users can set a priority (high, medium, low) when adding a task.

3. \*Due Dates\*: Users can set a due date for each task.

4. \*List View\*: Displays tasks with their status, priority, and due date (list\_tasks).

5. \*Data Persistence\*: Tasks are saved to and loaded from a JSON file (load\_tasks, save\_tasks).

**Running the Application**

To run the application, simply execute the script:

sh

python todo.py

This setup should provide a functional command-line to-do list application with the specified features.