**Task Management Command Line Application**

**Introduction**

An essential component of both personal and professional productivity is task management. The purpose of the Task Management (CLI) Application is to give users an intuitive and adaptable command-line interface (CLI) for effective task management. This study examines the Task Manager Application's architecture, features, functionality of the main menu, unit testing, and possible improvements.

**Why Python as Language Used?**

Because of its popularity and versatility, Python is a great option for building the Task Manager Application. Python stands out for its readability and simplicity, which are made possible by a clear syntax that improves code comprehension. This feature is very helpful when it comes to the Task Manager Application since clear and uncomplicated code makes the user experience more pleasant.

Python's fast development reputation is another important benefit. Its wide standard libraries and clear syntax make it easier to develop features and ensure that upgrades and enhancements are made on time. The extensive library and framework ecosystem of the language increases development efficiency even more.

**Concepts Utilized while Development Task Management System**

Among the ideas that are employed in the Task Manager Application, Object-Oriented Programming (OOP) is the most prominent. The programme models tasks as objects by utilising OOP principles. The OOP ideas of encapsulation and abstraction are best shown by the Task class, which contains characteristics like name, description, priority, category, and due date. The TaskManager class demonstrates the potential of object-oriented programming (OOP) in structuring and organising code by efficiently managing a set of tasks. One important component of the application's architecture is modular design. Every function, including adding tasks, tagging tasks as finished, and finding tasks, is contained within distinct methods. With each piece of functionality segregated and easily expandable, this modular approach improves the readability, maintainability, and organisation of the code.

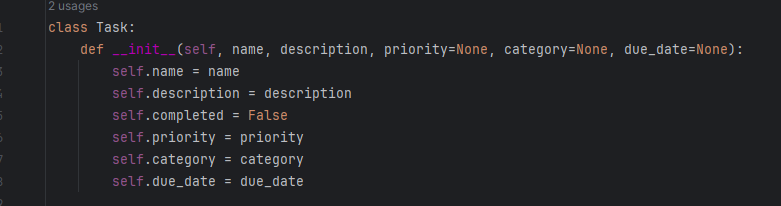
The concepts of menu navigation, answer creation, and user input processing form the foundation of the application's command-line interface (CLI) interaction. To give customers an engaging and smooth experience while navigating through the main menu's numerous options, iteration and control flow concepts are essential. Unit testing must be implemented if the application is to be reliable. The robustness of the codebase is increased by using the unit test module to generate test cases that verify the accuracy of functions. This dedication to unit testing guarantees that every feature works as intended, enhancing the application's overall quality. Furthermore, the application's primary menu feature effectively utilises control flow and iteration. Users are guided through a variety of alternatives by loops, conditionals, and user input processing, which then carry out the necessary capabilities based on their selections. An intuitive user experience is enhanced.

To sum up, Python is a great choice for creating the Task Manager Application because of its readability, ease of use, robust ecosystem, and support for object-oriented programming. Utilising fundamental programming ideas like OOP, modularity, and control flow, the application offers a dependable and approachable task management solution.

**System Architecture**

**Task Class**

The Task class, which encapsulates the characteristics and actions of distinct tasks, forms the basis of the programme. The name, description, priority, category, due date, and completion status of each task object identify it. Code organisation and modularity are facilitated by the object-oriented approach, making extension and maintenance simple. The task class code snapshot is shown in Fig. 1.



Figure

**TaskManager Class**

The main part in charge of overseeing a group of tasks is the TaskManager class. The methods for adding, viewing, deleting, prioritising, searching, categorising, assigning deadlines, and locating tasks are all contained in this class. Within the TaskManager instance, the tasks are kept as instances of the Task class in a list.  
  
**Main Functionality**

The primary purpose of the application serves as the orchestrator for its core functionality. This function shows users a menu of options while running in a loop. Entering the appropriate option number allows users to engage with the application and conduct a variety of operations with ease.

**Main Menu Functionality**

The centre of user interaction is the main menu function (main). It processes the selections that users make after presenting them with a menu of possibilities. The following choices are available on the menu:

1. Add New Task
2. View All Tasks
3. Mark Task as Completed
4. Delete Task from Available Task List
5. Prioritize Any Task
6. Search Task
7. Categorize Task
8. Set Due Date for a Task
9. Quit Task Manager

Until the user chooses to quit the application, the main menu function will keep asking them what they want to do. This is done using a loop. Users can easily use the programme thanks to its intuitive and modular design.

**System Features/ Function**

1. **Adding New Task**

* New tasks can be added by users, complete with a name, description, priority, category, and deadline.
* A new Task instance is created and added to the list of tasks via the TaskManager classes add\_task function.

1. **Viewing All Tasks**

* All tasks are shown in full view in the programme, together with information about their categories, titles, priorities, statuses, and due dates.
* The view\_tasks method loops through the task list and shows the user the pertinent data.

1. **Marking Task as Completed**

* Tasks can be marked as finished by users, changing their state from pending to done.
* The mark\_completed\_task function changes the completion state of a task by looking up the task by name.

1. **Deleting Tasks**

* By giving the task name to be deleted, tasks can be eliminated from the task list.
* The designated task is removed from the list using the delete\_task method.

1. **Prioritizing Tasks**

* Tasks can be ordered by the user by designating a priority level (High, Medium, or Low).
* The prioritize\_task function modifies a given task's priority.

1. **Searching for Tasks**

* Users can look for tasks whose names contain a particular term.
* If no matches are discovered, the search\_tasks method displays a notice and delivers a list of tasks that match.

1. **Categorizing Task**

* By allocating them to a certain category, tasks can be categorised.
* A task's categorization can be updated using the categorize\_task method.

1. **Setting Due Dates**

* By entering the task name and the new deadline in the format YYYY-MM-DD, users can set due dates for their tasks.
* The supplied task's due date is updated by the set\_due\_date method.

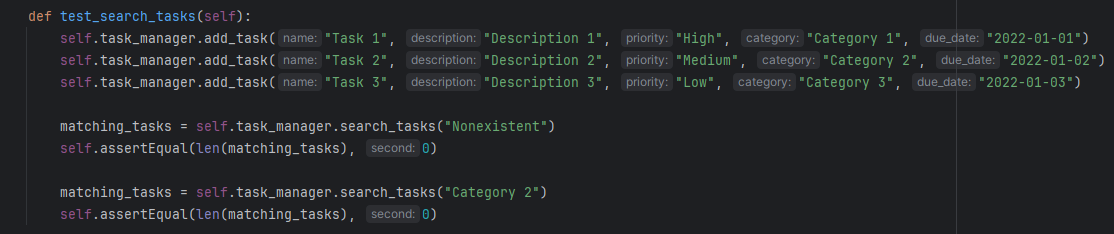
1. **Exiting the Application**

* The "Quit Task Manager" menu item allows users to end an application gracefully.

**Unit Testing**

To ensure the Task Manager Application's stability and dependability, unit testing is essential. The purpose of the unit tests is to confirm that each application method operates as intended.

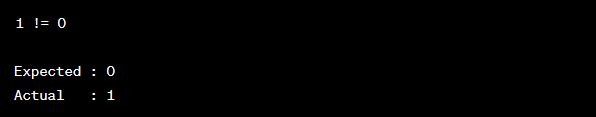
**Test Case Example: Searching for Task**

****The unit test in this case verifies that the search\_tasks method correctly finds tasks that include a given keyword. By ensuring that the term and category are both present, it offers a strong validation of the search feature.

Figure

**Handling Test Failures**

The error message gives comprehensive details about the expected and actual results if the test fails. As a shown in Fig.3.



Figure

This data helps developers identify the problem's origin and implement the required fixes to preserve the integrity of the application.

**Potential Enhancements**There are many opportunities for future improvements, and the Task Management System provides a solid base for effective work management. First and foremost, the implementation of user authentication is essential for guaranteeing the safe handling of task data and enabling customised task tracking for several users. Task dependencies, which let users define requirements or dependencies between tasks for improved project management, are another way the application might advance. A proactive component might be added by putting in place a reminder system, which would alert users to approaching deadlines or incomplete assignments. Integrating data persistence techniques, like file storage or database integration, would improve data management and offer a more dependable and scalable solution. More sophisticated search features, such as date ranges and logical operators, could significantly increase task retrieval accuracy.Giving users the option to select and categorise jobs according to different criteria improves their organisational skills. Not only would a graphical user interface (GUI) improve the aesthetic appeal, but it would also make the user experience more interactive. Teamwork can be promoted by collaborative features that let users share duties and work together on projects. Finally, integrating analytics into work distribution, productivity trends, and job completion rates provides insightful information that enhances task management.

**Conclusion**

Through a command-line interface, the Task Management System provides a flexible and easy-to-use solution for efficient task management. A strong and dependable user experience is guaranteed by its modular architecture, large feature set, and unit testing. The application has the potential to grow into a comprehensive task management tool that meets a wider range of user needs with future development and additions. The application is more maintainable and has higher quality assurance thanks to the included unit tests and thorough error reporting. With the increasing adoption of task management solutions by users, the Task Manager Application is prepared to expand and adjust to meet their changing needs.