**Introduction**

Task management is a crucial part of the contemporary life, as students, professional, and households have to cope with the tasks. One of the most popular tools to use in this purpose is a to-do list as it will enable the user organise tasks, give priorities, and monitor the progress. As technology has taken over, the paper lists have been substituted by the computer tools which are more convenient and adjustable. The purpose of this project was to design and develop a Python-based smart to-do list programme that would give users an effective method of recording a task besides including extra motivation in the form of quotes obtained via an external application programming interface (API).

The idea to create this project was attributed to the necessity to have a simple but useful task manager that might be employed by students and working people. There are a lot of to-do applications that are either too complicated or too simple. The Smart To-Do List that was created within the framework of the given project features the application of object-oriented programming, file processing with the help of JSON, exception management, the graphical user interface (GUI) that was developed with Tkinter, as well as the API integration. Combinations of these features result in an effective and easy to use tool that takes the position of task management in real world and satisfies the terms of the coursework. The fact that motivational quotes are added to the API is an emotional boost to the users who also feel positive about the application due to this stimulation ( API Ninjas, n.d.).

**Project Objectives and Requirements**

The main aim of this project was to develop a basic and useful task management system in Python 3. The course requirements directed the project, and they were object-oriented design, handling files, exception handling, and integrating API.

* The objectives of the project, in particular, were to:
* Give users an option of adding tasks with the help of natural language input.
* Split the user input by regular expressions in order to extract useful information like description and date of tasks.
* Persist tasks with a JSON file.
* To make it user friendly, develop GUI using Tkinter in a clean and feasible way.
* Add inspirational quotes API such that a user can see an inspirational quote when he/she is working on his/her duties.

It should be noted that the choice of the JSON file storage option was due to the fact that this data format is lightweight, human-readable, and easily supported in Python (Python Software Foundation, 2025a). The GUI will be written in Tkinter because it comes with Python by default and is therefore easily available to students and developers (Python Software Foundation, 2025b). The input was processed using regular expressions since they offer an easy and efficient means of analysing the patterns of natural language (Python Software Foundation, 2025c). Lastly, API Ninjas Quotes API was chosen due to its free nature, dependability, and feasibility ( API Ninjas, n.d.).

**System Design and Architecture**

The system was written in the object-oriented manner such that it could be easily maintained and was also designed to be modular. The app consists of four major units each in form of a class.

Task class The Task class is a single task. It contains description, due date and status of completion. The TaskManager class deals with the set information of tasks, the addition, deletion, saving, and loading of the tasks in the JSON file. QuoteAPI class connects with Quotes API to retrieve inspirational quotes. Lastly, the GUI class will develop the user interface through Tkinter and will connect all the parts.

The components interact with one another easily. Once the user input is typed on the GUI, regular expression is used to scan the entry in order to extract any date or special keywords. TaskManager creates and stores the Task object. TasksManager then stores the tasks as a JSON file to enable their session-independence. Meanwhile, the QuoteAPI retrieves a new inspirational quote, and it is shown in the interface.

This modular design is quite convenient in that it simplifies the code as well as extensibility. As an illustration, in future the addition of new features like categorisation of tasks or integration of calendar can be made without altering the whole system (Shipman, 2010).

**Implementation Details**

This capability to receive natural language was a major attribute. As an example, a user may type in submit assignment tomorrow and the system would take submit assignment as description and tomorrow as due date. This was done by re library of Python of regular expressions (Python Software Foundation, 2025c). Although regex is not able to resolve the complexity of a natural language, it was adequate in simple patterns such as days of the week or dates.

Motivational quotes are previously stored in the Quotes API to enable the experience to be better. The app submits a request to the API endpoint and receives a response in JavaScript, which is a quote with randomly accessible information. The quote will then be presented in the upper part of the GUI, offering inspiration every time the user would be using the app (API Ninjas, n.d.). Exception handling is used to make sure that in case the API is not available the programme does not crash a fallback message is provided.

Tasks were stored using JSON due to its simplicity and the fact that it intertwines with Python. Every job is written as a dictionary and added into a list, which is subsequently written to a JSON file. On programme initiation, the file is again read into the application to recover tasks. This enables the users to exit and reopen the application without any data loss (Python Software Foundation, 2025a).

Exception handling was well incorporated in the project. An illustration is when the JSON file is lost or damaged, another one is generated by the programme. In the case where the API call fails, the user will be notified with an error message, and the programme will not end abruptly. Likewise, invalid user entries are also dealt with decently, by displaying a message within the GUI.

The GUI was constructed based on Tkinter and the ttk modern Tkinter using the modern ttk widgets to enhance usability and beauty. The interface includes text entry field of actions, adding and removing task buttons, and the list view to show the existing actions. The top of the window displays a motivational quote, which makes the application look interesting to the eyes. The design-related decisions were based on the Tkinter development tips and techniques (Real Python, 2025; TkDocs, 2025; Geeksforgeeks, 2025a).

**Challenges and Solutions**

Natural language parsing was one of the major issues. Although the regular expression tool is helpful, it cannot work with unclear input. It was managed by only allowing common patterns including dates and days (Python Software Foundation, 2025c).

The other issue was the integrity of the JSON file. The file may be corrupted in case the programme ended up being corrupted in the writing stage. In this regard, the tasks were written in Python through the built-in Python JSON module that guarantees the structured format (Python Software Foundation, 2025a).

The reliability of API was also a problem because the application relies on the internet. The workaround was to have exception handling of poor API calls, so that the user can still get a fallback message when there is no quote (API Ninjas, n.d.).

Lastly, it was not easy to design a clean and user-friendly GUI. Although powerful, Tkinter offers little choice on the style. In order to defeat this, the references to Tkinter were followed with the help of modern ttk widgets and layout managers (Shipman, 2010; TkDocs, 2025).

**Testing and Evaluation**

A number of scenarios were tested on the application. The provision of tasks having a definite description was effective, and idiomatic due dates were extracted through the use of regex. To ensure that data are preserved, saving and loading of tasks to JSON was tested according to closing and reopening a programme.

Even the API integration was tested by making sure that a new quote was displayed whenever one started an application. Disconnection to the internet was also simulated so as to ensure that error handling was effective. Informally, the GUI was tested in terms of usability. There were correctly answering buttons, well displayed tasks and quotations provided a positive touch.

In general, the application fulfilled all the functional requirements, but it could be enhanced in natural language processing and features such as task prioritisation.

**Conclusion and Future Work**

The Smart To-Do List Project was a successful endeavour that incorporated the principles of Python programming with the application development. It showed the lesson using object-oriented programming, file manipulation based on JSON, exception handling, developing a GUI using Tkinter, and integrating an API. The end product is an effective task management tool that does not only assist the user to arrange their duties, but it also motivates them to get work done by providing quotes.

The limitations of the use of regexes and GUI design were also highlighted in the project, which was handled by thorough designing and error management. Learning wise, the project presented a rich experience in understanding how various concepts of programming can be incorporated to form a practical application in the real world.

It might be enhanced by incorporating better natural language processing functionality, using it with Google Calendar, task-centric approach, or a mobile application. Such extensions would make the applications more functional and interesting to more people.

To sum up, this project completed the coursework task and also produced a valuable resource which shows the usefulness of Python and APIs in the resolution of various daily tasks.

**References:**

API Ninjas (n.d.) *Quotes API*. Available at: https://api.api-ninjas.com/v1/quotes (Accessed: 21 September 2025).

GeeksforGeeks (2025a) *Python Tkinter Tutorial*. Available at: [https://www.geeksforgeeks.org/python/python-tkinter-tutorial/](https://www.geeksforgeeks.org/python/python-tkinter-tutorial/?utm_source=chatgpt.com) (Accessed: 21 September 2025).

GeeksforGeeks (2025b) *Regular Expression in Python (with examples)*. Available at: [https://www.geeksforgeeks.org/python/regular-expression-python-examples/](https://www.geeksforgeeks.org/python/regular-expression-python-examples/?utm_source=chatgpt.com) (Accessed: 21 September 2025).

Python Software Foundation (2025a) *json — JSON encoder and decoder*. Available at: https://docs.python.org/3/library/json.html (Accessed: 21 September 2025).

Python Software Foundation (2025b) *tkinter — Python interface to Tcl/Tk*. Available at: [https://docs.python.org/3/library/tkinter.html](https://docs.python.org/3/library/tkinter.html?utm_source=chatgpt.com) (Accessed: 21 September 2025).

Python Software Foundation (2025c) *re — Regular expression operations*. Available at: [https://docs.python.org/3/library/re.html](https://docs.python.org/3/library/re.html?utm_source=chatgpt.com) (Accessed: 21 September 2025).

Python Software Foundation (2025d) *Regular Expression HOWTO*. Available at: [https://docs.python.org/3/howto/regex.html](https://docs.python.org/3/howto/regex.html?utm_source=chatgpt.com) (Accessed: 21 September 2025).

Real Python (2025) *Python GUI Programming: Your Tkinter Tutorial*. Available at: [https://realpython.com/python-gui-tkinter/](https://realpython.com/python-gui-tkinter/?utm_source=chatgpt.com) (Accessed: 21 September 2025).

Shipman, J.W. (2010) *Tkinter 8.5 reference: a GUI for Python*. Available at: [https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html](https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html?utm_source=chatgpt.com) (Accessed: 21 September 2025).

TkDocs (2025) *Tutorial — Graphical User Interfaces with Tk*. Available at: [https://tkdocs.com/tutorial/](https://tkdocs.com/tutorial/?utm_source=chatgpt.com) (Accessed: 21 September 2025).