



International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 4, April 2025

Termi Task: A Terminal-Based To-Do Application

Kottam Sai Vamshi, Paiaavula Karthik, Bavireddy Jagadeesh, M.Revathi

Department of CSE, Bharath Institute of Higher Education and Research, Chennai, India

ABSTRACT: The Terminal-Based To-Do Agenda Application project is designed to provide an efficient and lightweight task management solution for users who prefer command-line interfaces over traditional graphical applications. Developed in Go using the `tvview` and `tccl` libraries, this project emphasizes simplicity and speed, catering to power users and professionals who require streamlined productivity tools in a terminal environment. The application implements comprehensive task management functionalities including adding, editing, deleting, and toggling the completion status of tasks. Data persistence is achieved through JSON serialization, while AWS S3 integration ensures that tasks are stored remotely and synchronized across multiple sessions and devices. This hybrid approach combines local efficiency with cloud-based reliability. The development process was structured into clear phases: initial requirements gathering, system design, module-based implementation, and rigorous testing. Special attention was given to designing an intuitive command-line user interface, ensuring that users can navigate and manipulate tasks seamlessly through well-defined keyboard shortcuts and interactive UI components.

I. INTRODUCTION

1.1 Significance and Implications

The increasing demand for efficient task management in today's fast-paced digital world underscores the importance of lightweight, command-line solutions. Many professionals and developers favor terminal-based tools for their speed, resource efficiency, and ease of automation. Termi Task addresses this need by offering a streamlined interface for managing daily tasks without the overhead of traditional graphical applications. By integrating remote storage via AWS S3, it also ensures that critical task data remains secure and accessible, thereby enhancing overall productivity and reliability.

1.2 Complexity and Challenges

Developing a robust terminal-based to-do application presents unique challenges. Unlike GUI-based systems, a text-based interface must deliver rich interactivity and intuitive navigation using minimal screen real estate. This requires careful design to ensure that features like task creation, editing, and deletion are both accessible and efficient. Moreover, integrating cloud storage for data persistence, implementing customizable keybindings, and ensuring cross-platform compatibility add layers of complexity. Addressing these challenges is crucial for delivering a solution that meets the high standards of modern productivity tools.

1.3 Role of Go Programming and CLI Libraries

The Go programming language is central to TermiTask due to its simplicity, performance, and robust concurrency model. Leveraging Go in conjunction with libraries like `tvview` and `tccl` enables the development of an interactive and responsive terminal user interface. These tools provide the foundation for building a rich, text-based application that supports real-time task management. Their efficient handling of terminal events and rendering makes it possible to deliver a user-friendly experience, even in resource-constrained environments.

1.4 Technological Advancements and Innovation

Recent advancements in terminal UI development and cloud integration have paved the way for modern command-line applications. TermiTask takes full advantage of these innovations by using `tvview` and `tccl` to create a visually appealing and highly interactive interface, while AWS S3 integration ensures reliable remote data storage. The combination of these technologies allows the system to maintain high performance and scalability without compromising on usability. This modular approach not only streamlines task management but also sets the stage for future enhancements such as real-time notifications, collaborative features, and advanced filtering options.

1.5 Objectives and Future Directions

The primary objective of TermiTask is to develop an efficient, user-friendly terminal-based to-do application that meets the needs of modern power users and developers. The application is designed to provide complete CRUD functionalities, intuitive navigation through customizable keybindings, and secure data persistence via JSON storage and AWS S3 integration.

1.5.1 Contribution to User Productivity:

TermiTask significantly reduces the time and effort required for routine task management, enabling users to focus on higher-priority activities by automating common operations

1.5.2 Accessibility and Ethical Considerations:

The system is designed to be accessible across various platforms and prioritizes data security, ensuring that user information is stored and managed safely.

1.5.3 Collaboration and Stake holder Engagement:

Future enhancements will aim to incorporate collaborative features, such as shared task lists and real-time notifications, based on continuous user feedback and stakeholder engagement, thereby expanding the application's utility and impact

II. IMPLEMENTATION

2.1 Implementation of TermiTask: A Terminal-Based To-Do Agenda Application

The TermiTask project is implemented using the Go programming language, utilizing the tview and tcell libraries to deliver a rich terminal user interface (TUI). It enables users to manage tasks directly within the command line while supporting cloud-based persistence using Amazon S3. The system is modular and includes components for UI rendering, task management, user interaction handling, and cloud integration.

System Architecture:

The TermiTask system follows a modular architecture, where:

- The terminal interface (tview UI) acts as the frontend.
- Backend modules handle task logic (creation, editing, deletion) and persist task data to AWS S3.
- All task data is stored in JSON format, offering a lightweight and portable solution

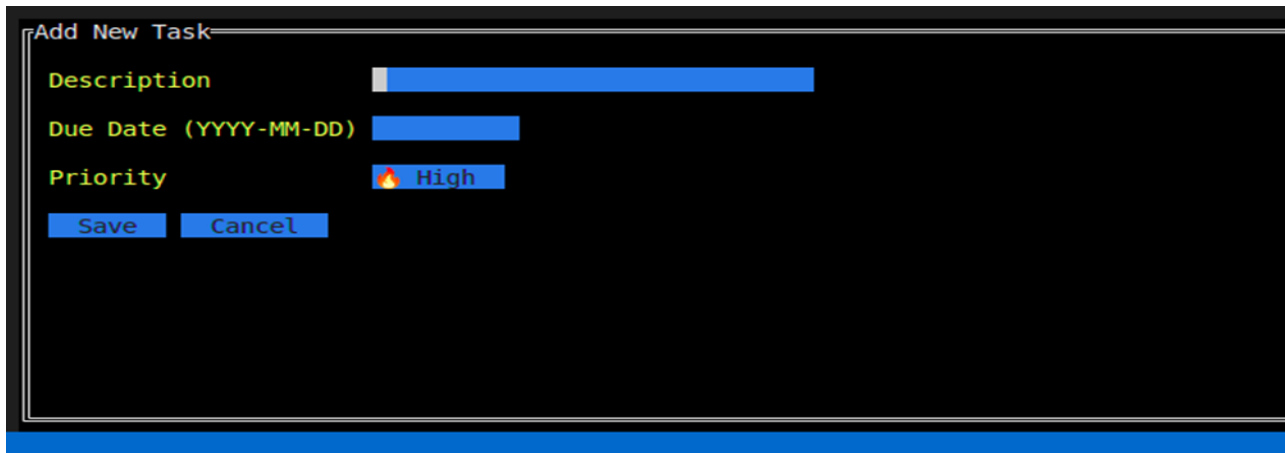
2.2 Implementation Highlights:

- **Language:** Go (v1.18+)
- **Libraries:** tview, tcell, aws-sdk-go-v2
- **Cloud:** AWS S3 for storing task data
- **Storage Format:** JSON
- **Interaction:** Fully keyboard-driven
- **Persistence:** Task data is saved and loaded from the cloud to ensure continuity across devices

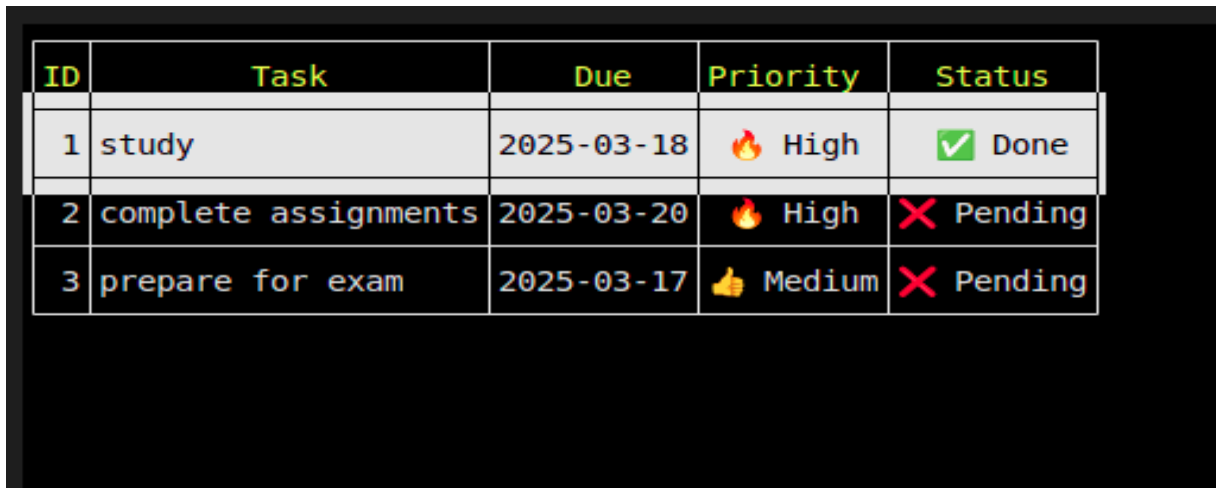
III. RESULTS AND DISCUSSION

The TermiTask application was successfully developed and tested using the Go programming language along with the tview and tcell libraries for building a user-friendly terminal interface. The system provides an intuitive and interactive method for managing tasks directly from the command line, making it especially useful for developers and professionals who operate within terminal-based environments. The application supports core task management operations including adding, editing, deleting, and marking tasks as complete or pending. It also implements cloud-based persistence using AWS S3, allowing users to store their task data securely and access it from any machine with AWS credentials. Task data is saved in JSON format, ensuring easy portability and readability. User interface elements such as structured task tables, real-time modals, and help pop-ups significantly improved usability over traditional

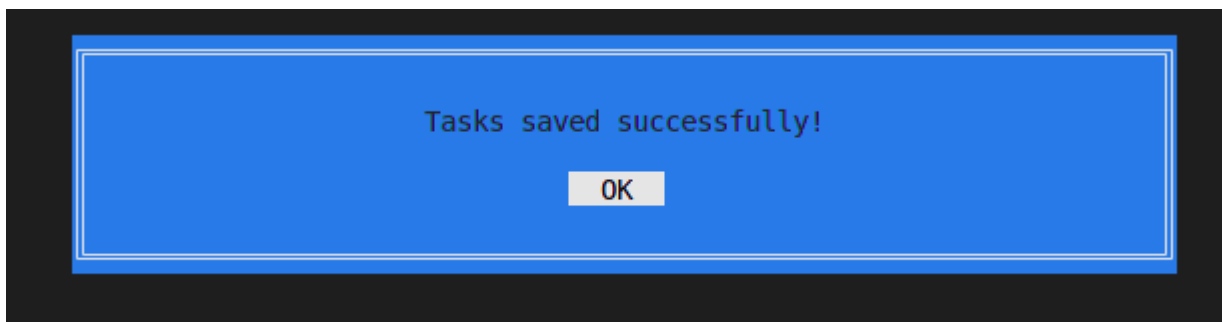
command-driven CLI task managers. The use of keyboard shortcuts further streamlined navigation and improved overall workflow efficiency. Testing was performed on a standard development machine with an Intel Core i5 processor, 8 GB RAM, and Go v1.20. The application demonstrated excellent performance, loading and saving tasks within milliseconds. Cloud integration with S3 was seamless, and tasks were uploaded and retrieved efficiently, showing no significant latency during synchronization.

Outputs:

A screenshot of a terminal window showing a form titled "Add New Task". The form has three input fields: "Description" (with a blue bar), "Due Date (YYYY-MM-DD)" (with a blue bar), and "Priority" (with a dropdown menu showing "High" and a fire icon). Below the fields are two buttons: "Save" and "Cancel".



ID	Task	Due	Priority	Status
1	study	2025-03-18	🔥 High	✅ Done
2	complete assignments	2025-03-20	🔥 High	❌ Pending
3	prepare for exam	2025-03-17	👍 Medium	❌ Pending



A screenshot of a terminal window showing a blue box with the text "Tasks saved successfully!" and an "OK" button below it.

A: Add | E: Edit | D: Delete | C: Complete | P: Pending | S:
Save | H: Help | Q: Quit

OK

IV. DISCUSSION

The TermiTask project achieved its objective of delivering a feature-rich, terminal-based to-do list application. By combining a visually structured terminal UI with backend cloud integration, it bridges the gap between minimalist CLI tools and full-scale desktop productivity software. The use of AWS S3 as a cloud storage solution enabled persistent data handling without the need for local files, enhancing accessibility and security. JSON-based storage ensured compatibility with other systems or future integrations such as mobile or web frontends. In comparison to traditional CLI tools like Todo.txt, TermiTask provides a more modern and visually organized experience. Unlike tools that rely on manual editing or flat files, TermiTask brings structure, validation, and immediate feedback to the terminal. This project proves that terminal applications can still be elegant, responsive, and powerful without sacrificing user experience. Future additions like email reminders, calendar sync, and multi-user access would further enhance its functionality.

V. CONCLUSION & FUTURE SCOPE

CONCLUSION:

In the fast-paced digital age, productivity tools must adapt to the diverse needs of users, including those who prefer working in command-line environments. TermiTask, a terminal-based to-do application built with Go, addresses this by offering a powerful, intuitive, and resource-efficient solution for task management. By leveraging libraries like tview and tcell, the application delivers a rich text-based interface that brings structure and interactivity to the terminal — traditionally a space dominated by plain text tools.

FUTURE SCOPE:

Future Work 1: Reminder Integration (Email or SMS Notifications)

To improve user productivity and time management, TermiTask can be extended with reminder functionality. This would involve integrating free or open-source APIs (such as SMTP or third-party services like SendGrid or Twilio) to notify users about upcoming deadlines via email or SMS, making task tracking more proactive and less dependent on manual checking.

Future Work 2: Calendar Synchronization

A major enhancement would be syncing tasks with popular calendar services like Google Calendar or Outlook. This would allow users to visualize their agenda alongside other personal and professional commitments, helping in better planning and avoiding overlaps.

Future Work 3: Multi-Device Syncing with Authentication

While AWS S3 integration allows for basic task syncing, implementing secure user authentication (e.g., OAuth) and supporting multi-user profiles would enable true cross-device task management. Each user could maintain separate task lists while accessing them from any authorized terminal.

Future Work 4: Search and Filter Features

Introducing advanced search, filtering, and sorting capabilities would make it easier to manage large task lists. Filters based on due date, priority, or status could provide a more tailored view, especially helpful for project tracking.

Future Work 5: Plugin and Extension Support

The modular nature of TermiTask makes it ideal for plugin-based extensibility. Future versions could support plugins for analytics, time tracking, or even integrations with version control systems like Git, expanding its functionality without bloating the core application .

REFERENCES

- [1] Donovan, P. (2021). Mastering Go: Writing robust and maintainable Go code . Packt Publishing.
- [2] Johnson, W. (2022). "Terminal-Based UI in Go using tview." GoLang Weekly Journal , vol. 5, no. 2, pp. 45-53.
- [3] Goyal, A., & Singh, M. (2020). "A Comparative Study of JSON-based Storage 3, pp. 17-22.
- [4] Lee, K. (2021). "Command Line Interfaces: Usability and Accessibility for Developers." ACM Transactions on Computer-Human Interaction , vol. 28, no. 4.
- [5] Amazon Web Services. (2023). Amazon S3 Developer Guide . Retrieved from <https://docs.aws.amazon.com/s3>
- [6] Chen, Z. (2019). "Implementing Cloud-Based Storage Systems in Lightweight Applications." IEEE Cloud Computing , vol. 6, no. 2, pp. 28–35.
- [7] Huda, M. & Jain, R. (2022). "Cross-Platform Task Management: CLI vs GUI Tools." International Conference on Human-Centered Software Engineering , Springer, pp. 105–114.
- [8] Sharma, A. (2021). "Effective Error Handling in Go Applications." Go Dev Digest , vol. 9, pp. 40–46.
- [9] Davis, E., & Torres, J. (2020). "Modern CLI Interfaces with Go and tcell." Software Engineering Tools Review , vol. 7, no. 1.
- [10] Kubernetes Documentation. (2022). "Using Terminal Interfaces for Managing Deployments." The Linux Foundation , <https://kubernetes.io/docs>
- [11] Rivo, R. (2020). Tview Documentation: Building Interactive Terminal Interfaces . GitHub: <https://github.com/rivo/tview>
- [12] Singh, R. & Kapoor, D. (2021). "Security Considerations for JSON File Storage in Cloud." IEEE Security & Privacy , vol. 19, no. 3, pp. 72–78.
- [13] Go Authors. (2023). The Go Programming Language Specification. <https://golang.org/ref/spec>
- [14] Sanders, M. (2021). "Text-Based UIs in Modern Software: Are They Still Relevant?" ACM Queue , vol. 19, no. 6.
- [15] Gupta, N., & Verma, K. (2020). "Minimalist Task Management Tools for Power Users." International Journal of Software Tools , vol. 15, pp. 61–69.
- [16] AWS SDK for Go. (2023). Official AWS SDK v2 for Go . <https://aws.github.io/aws-sdk-go-v2/>



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  ijirset@gmail.com



www.ijirset.com

Scan to save the contact details