Carleton University

ToDo O'clock

1. Project overview

To address the common challenge of efficiently managing schedules in our fast-paced world, I decided to create this program, ToDo O'clock. With busy work, study, and social schedules, it's easy for events to slip through the cracks or for users to feel overwhelmed by their commitments. I wanted to develop a solution that would empower users to better organize their time, stay on top of their tasks, and ultimately lead more balanced and productive lives. By creating ToDo O'clock, I aimed to provide users with a user-friendly and highly flexible schedule management program that would help them plan ahead and manage their schedules more efficiently.

In developing ToDo O'clock, I utilized several libraries and methods to implement the desired functionality. The core of the program was built using Python's Tkinter library for creating a graphical user interface (GUI). Tkinter provided the foundation for designing an intuitive and visually appealing interface, enabling users to interact with the program seamlessly. Additionally, I incorporated the tkcalendar library to integrate a calendar widget into the GUI, allowing users to select dates and navigate their schedules easily. For managing date and time-related operations, I leveraged Python's datetime library, which provided robust functionality for handling date inputs, event scheduling, and exporting schedules.

In terms of code structure and architecture, I followed a modular and organized approach to ensure readability, scalability, and maintainability. The program was divided into separate modules or classes for different functionalities, such as event management and GUI design. This modular structure allowed for easy navigation and modification of code components, enhancing the program's overall flexibility and extensibility.

In terms of code structure and architecture, I followed a modular and organized approach to ensure readability, scalability, and maintainability. The program was divided into separate modules or classes for different functionalities, such as event management, GUI design, and PDF generation. This modular structure allowed for easy navigation and modification of code components, enhancing the program's overall flexibility and extensibility.

One of the most interesting and creative aspects of ToDo O'clock is the welcoming window that greets users upon launching the program. This initial interaction sets a positive tone and engages users from the start, making them feel welcome and excited to use the program. Additionally, the user-friendly interface design, with intuitive navigation buttons and detailed event information, enhances the overall user experience and makes scheduling tasks a breeze.

Another creative aspect of the program is the implementation of a final executable (exe) program using PyInstaller. This feature allows users to easily install and run ToDo O'clock on their computers without requiring any additional setup or dependencies. By providing a standalone executable, users can

access the program conveniently and enjoy its full functionality without any technical hurdles.

In future iterations of ToDo O'clock, I plan to explore several additional features and functionalities to further enhance the program's usability and utility. Some potential features include:
1) Reminders and notifications: Implementing reminders and notifications to alert users of upcoming events or deadlines, helping them stay organized and on track.
2) Customization options: Providing users with customization options for the interface, event categories, color schemes, and other preferences to tailor the program to their unique needs and preferences.
3) Cloud synchronization: Integrating cloud synchronization capabilities to sync schedules across multiple devices and ensure data consistency and accessibility from anywhere.
4) Integration with other tools: Offering integration with other productivity tools, such as calendars, task managers, and project management platforms, to streamline workflow and enhance productivity

2. Objectives

In my final code, the main objectives include:
1) Creating Event Objects: The Event class is responsible for creating event objects, each consisting of a date, details, and completion status.
2) Managing Events: The Event list class handles the management of events. It allows users to add events, toggle completion status, and export event data.
3) User Interface Design: The program aims to provide a user-friendly interface for interacting with events. It utilizes Tkinter to create windows, frames, buttons, and text widgets, ensuring a visually appealing and intuitive user experience.
4) Data Import and Export: The File Import Window class enables users to import event data from a file, while the Event list class facilitates exporting in-progress events to a text file.
5) Integration and Functionality: The App class serves as the main application controller, orchestrating interactions between different components. It initializes the Tkinter window, manages UI transitions, and integrates the file import functionality with the event list.

The difference between the objectives in the final code and the initial proposal lies in the approach to managing events. While the proposal initially focused on a calendar-centric design with separate edit and view modes, the final code adopts an event-centric approach. This shift in focus led to the creation of the Event list window, where users can seamlessly add, view, and manage events in a unified interface.

Additionally, the final code places greater emphasis on user interaction and interface design, aiming to provide a more intuitive and streamlined experience. The integration of features such as checkbuttons for toggling event completion status and dynamic event list updates enhances usability and functionality.

3. Challenges

Refining the program's design and functionality proved to be the most challenging task during development. Initially, I aimed to implement two distinct modes: edit and view. In the edit mode, users could input events into specific dates within a monthly calendar, while the view mode allowed for an overview of all added events and provided detailed information for specific dates. Additionally, users could mark events as completed and export data for ongoing tasks.

However, as I delved into the code, I encountered challenges with the integration of various functions within the edit and view modes. The original structure was muddled, with both modes sharing similar functions. To streamline the code and improve readability, I opted to refactor the design by introducing specific functions to replace the edit and view modes. This approach helped clarify the code logic and enhance overall comprehensibility.

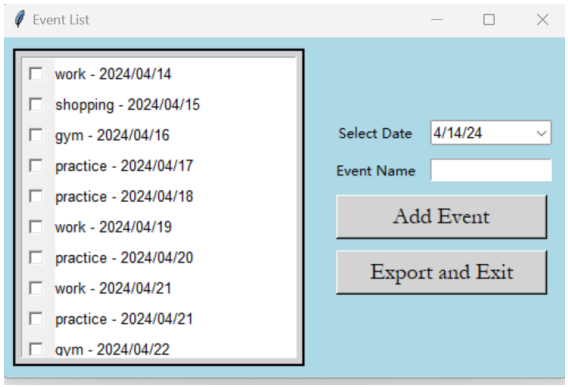
Furthermore, I faced hurdles in effectively accessing the calendar module. Upon reassessment, I realized that the heart of the program lay not in the calendar itself, but rather in managing events. Thus, I shifted gears towards an event-centric approach, resulting in the creation of the Eventlist window. By consolidating functions such as adding events, viewing events, modifying statuses, and exporting data into a unified interface, I synthesized the core functionalities outlined in the proposal. This strategic pivot facilitated a more cohesive and user-friendly experience, closely aligning with the program's objectives and meeting user needs.

4. Observations

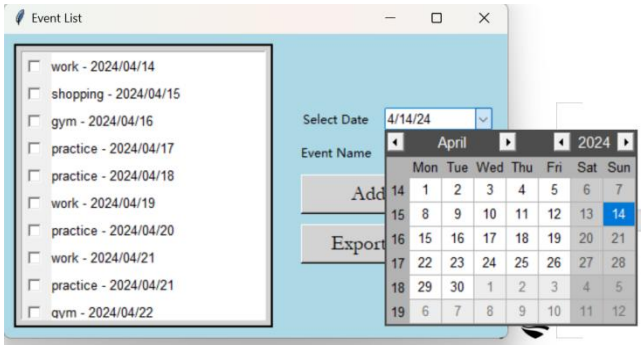
When users open TODO O'clock, they will be greeted with a welcoming window, With a simple click on the Explore button, users can seamlessly move to the next step.

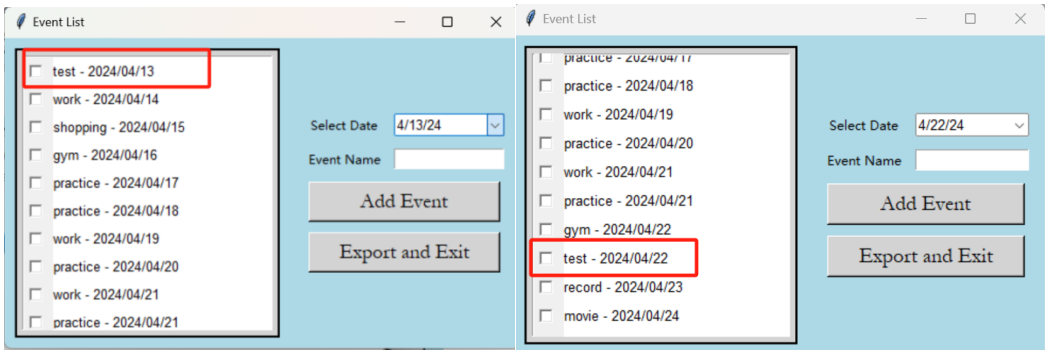
Then, in the File Import Window, users can import local data, ensuring continuity in managing events.

After importing the data, the Event list Window appears, all the imported events will be displayed here. On the right side, users have the option to add new events or export data.

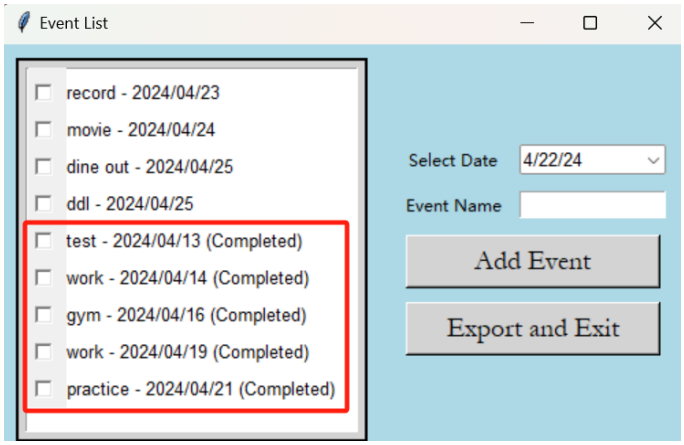


To add a new event, simply select a date using the combobox and enter the event details. Then, click the Add Event button, and the event will be seamlessly added to the list. all events are sorted by date, whether they are from local data or newly added.





when an event is completed, users can click the checkbox, it will automatically move to the bottom of the list and displays (completed)



When users are done, they can save progress by clicking the Export and Exit button. After users save the data, they can seamlessly exit from the program.