

ENGM 4620 Python for Engineers

Project #1: Daycare Management System

Prepared For:

Issam Hammad (Instructor)

Jack Langille (TA)

Prepared by:

Jonathan Chartrand

B00365615

Robert Boutette

B00882311

February 24, 2024

Table of Contents

1.0	Concept	2
2.0	Project Development	2
3.0	Project Description	3
3.1	Database Management UI	3
3.2	Payment Management System	4
3.3	Calendar Feature	5
4.0	Error Handling/Fine Tuning.....	7
4.1	Database Integrity and Validation	7
4.2	Operational Constraints	7
4.3	User Interface Feedback.....	7
5.0	Conclusion.....	8

List of Figures

Table 1 - Individual Contribution by stage.....	3
Figure 1 - Main Menu GUI.....	3
Figure 2 - View Database window.....	4
Figure 3 - Add and Remove Child windows.	4
Figure 4 - Payment window.	5
Figure 5 - Calendar and Attendance windows.....	6

1.0 Concept

The purpose of this project is to develop a simplified daycare management system specifically tailored for at-home daycare use. The idea to design a daycare management system for this project stemmed from Robert's experiences at home, as his wife runs an in-home daycare. The focus is on creating a practical and functional tool that caters to the essential aspects of running a daycare, such as managing child enrollments, organizing daily schedules, managing a database, and processing payments.

2.0 Project Development

Initially, our project was envisioned as a basic command prompt-based system, primarily because of our limited experience with Python. The plan was to manage the daycare operations by reading from and writing to CSV files for data storage and using a simple input-based menu to navigate through the system. This approach was straightforward, relying on text inputs and outputs to handle tasks like tracking enrollments, scheduling, and payments.

However, our project took a significant turn when we started using GitHub Copilot. With its assistance, we quickly learned how to incorporate Python widgets through Tkinter, enhance our system with a calendar for scheduling, and improve the interface using the Treeview widget from the ttk module. Tkinter allowed us to develop a graphical user interface (GUI), making the system more user-friendly and interactive than the original command-line concept.

The calendar widget helped us in implementing a visual and interactive calendar, making it easier to manage daily schedules and appointments. Meanwhile, ttk.Treeview provided us with the ability to display the stored data in a more organized and accessible manner, such as lists of enrolled children and their details.

This shift from a command-line interface to a GUI-based system, powered by our exploration of Tkinter and other Python libraries with the help of GitHub Copilot, transformed our project into a more sophisticated and practical tool for daycare management. This journey not only significantly improved the functionality and usability of our project but also accelerated our learning curve in Python, showcasing the power of leveraging the right tools and resources to expand our technical capabilities.

3.0 Project Description

This section outlines the chronological development and implementation of features within our daycare management software project. We will explore the sequence in which the system's capabilities were added, the rationale behind key decisions made throughout the development process, and the technical solutions employed to address the project's requirements. Individual contributions for the project and the project stages can be seen in Table-1. Brainstorming and error handling were a collaborative effort.

Table 1 - Individual Contribution by stage

Member	Database Management UI	Payment Management System	Calendar Feature
Jonathan	- Main menu GUI - Top level window GUI	- Window GUI - Feedback mechanism	- Calendar UI - Attendance Management
Robert	- View/Modify Database (Add/Remove)	- Process payments - Modify database	- Backend Integration

3.1 Database Management UI

As the initial module in our application, the Database Management User Interface (UI) was pivotal. We aimed for a design that could streamline the administrative tasks of a home daycare, centralizing the management of child enrollment data. Our choice to use Tkinter as the graphical user interface (GUI) framework was driven by its Pythonic integration, ease of use, and wide support within the Python community. This decision enabled us to focus on implementing practical features rather than grappling with the intricacies of GUI programming.

Main Menu (Figure 1): The Main Menu serves as the gateway to the application's functionality. With a layout constructed using Tkinter's Frame and Button widgets, it provides a straightforward navigation path to all the major features of the system. We were deliberate in our approach to ensure the UI's simplicity to accommodate users with varying degrees of technical proficiency.

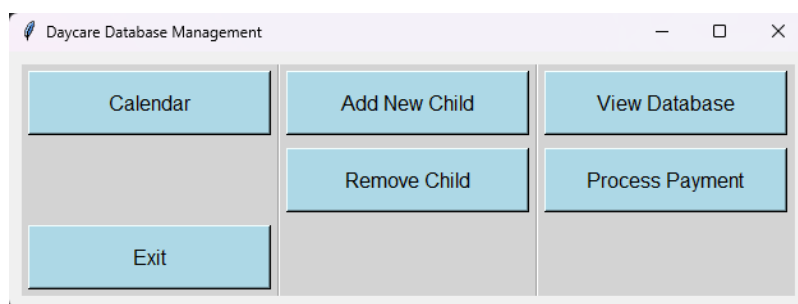
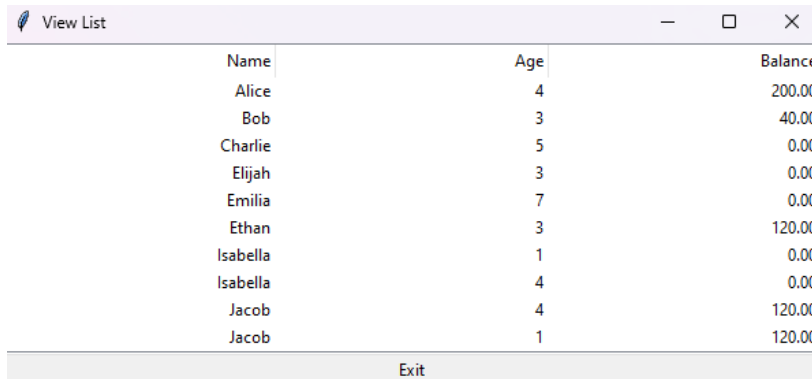


Figure 1 - Main Menu GUI.

View Database (Figure 2): The "View Database" button employs Tkinter's ttk.Treeview widget, allowing us to display children's enrollment data in an organized tabular format. The widget's functionality was utilized to present information in a structured manner, enhancing the system's usability.



Name	Age	Balance
Alice	4	200.00
Bob	3	40.00
Charlie	5	0.00
Elijah	3	0.00
Emilia	7	0.00
Ethan	3	120.00
Isabella	1	0.00
Isabella	4	0.00
Jacob	4	120.00
Jacob	1	120.00

Exit

Figure 2 - View Database window.

Add New Child (Figure 3): The feature to add a new child leverages Tk's Entry widgets within a Toplevel window, a dialog that pops up over the main application window. This separate interface reinforces our system's data integrity by ensuring the name field accepts only alphabetic input and the age field is limited to positive integers. These validations are crucial for maintaining the accuracy of the daycare's records.

Remove Children (Figure 3): The "Remove Child" feature is designed with user convenience in mind. An OptionMenu widget dynamically populates with the names of enrolled children, streamlining the removal process. This dropdown menu ensures that users can efficiently manage the daycare's roster with minimal interaction cost.

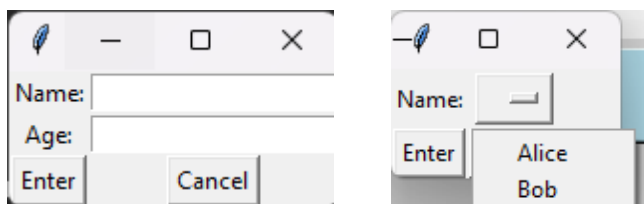


Figure 3 - Add and Remove Child windows.

3.2 Payment Management System

Following the establishment of the Database Management UI, our attention shifted to the Payment Management System. As the next step in our project's progression, we sought to integrate the balance management feature tightly with the existing database, highlighting the system's capability to handle

file management and data modification. Utilizing Tkinter, we aimed to deliver a module that allows for financial transactions and that demonstrates file manipulation by modifying the database after payment.

Processing Payments: To apply a payment, the user interacts with the 'Process Payment' button, located in the Main Menu. This triggers a Tkinter Toplevel widget, which unfurls a new window over the main interface. Here, the user is greeted with two Entry widgets. The first Entry field awaits the selection of a child's name, facilitated by an OptionMenu, which smartly populates with the names of children bearing a balance above zero. This ensures the application's logic remains targeted and efficient. The second entry field is where the payment amount is entered.

Applying Payments: Upon entering a payment amount, the user is expected to commit the transaction by clicking 'Apply'. The program then engages in a validation check, ensuring that the amount entered does not surpass the existing balance. In the event of an overpayment, the system is designed to adjust the balance to zero automatically and compute the excess funds to be returned, demonstrating the application's capability to handle unexpected user input gracefully.

Feedback Mechanism: Consistent with the principles of robust application design, the system furnishes immediate feedback through the messagebox module. Whether confirming the success of a transaction or flagging an entry error, the system communicates with clarity and precision, reinforcing user confidence and maintaining the integrity of financial records.

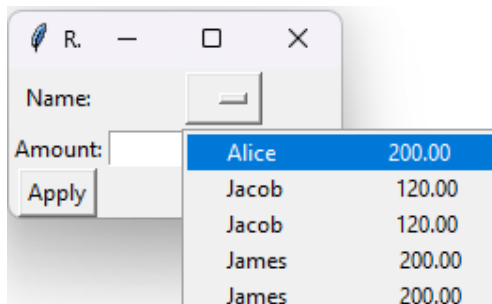


Figure 4 - Payment window.

3.3 Calendar Feature

The Calendar Feature, initially a secondary consideration in our development process, quickly became a focal point once we discovered the ease of implementing the tkcalendar widget. What started as a probing concept evolved into a key component of our system, enhancing the overall functionality and

user experience. Leveraging the tkcalendar and tkinter libraries, we were able to introduce a visually intuitive calendar within the graphical user interface (GUI).

Interactive Calendar UI: Utilizing the tkcalendar.Calendar widget, our application presents a navigable monthly calendar view, as seen in Figure 5, allowing users to select dates with a simple click. This integration is central to managing daily schedules, providing a visual and intuitive means to handle the complexities of daycare attendance.

Attendance Management: By selecting a date and pressing the 'Edit Date' button, the AttendanceWindow class, instantiated from attendance_window.py, is called. This brings up a new window where users can add or remove a child's attendance using a dropdown menu created with Tkinter's OptionMenu widget. This directly manipulates the attendance data stored in the backend, showcasing the seamless interplay between the front-end UI and the system's logic.

Backend Integration: The 'End Month' button initiates a backend process that calculates attendance fees, updates balances, and prepares a CSV report for administrative use. This demonstrates the intricate integration of frontend and backend operations, emphasizing the system's capability to manage complex tasks such as date manipulation and data persistence.

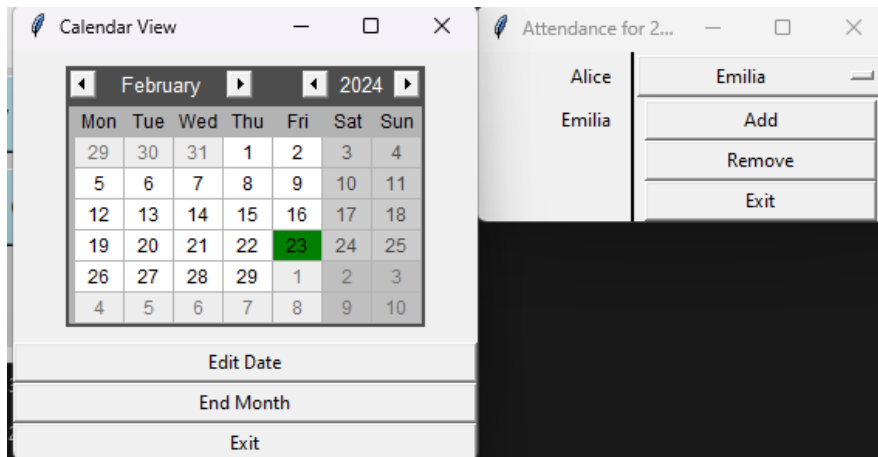


Figure 5 - Calendar and Attendance windows.

4.0 Error Handling/Fine Tuning

This section details the strategies and mechanisms we employed to manage errors and provide clear feedback to users, thereby enhancing the overall robustness of the system.

4.1 Database Integrity and Validation

Unique Name Check: In the `add_child` function, we introduced a check to ensure that each child's name is unique within the database. This is accomplished by invoking the `is_name_unique` method from the `DatabaseManager` class before adding a new entry. If a duplicate name is detected, the application alerts the user with a descriptive error message, preventing the addition of the child to the database and maintaining data integrity.

Data Type and Value Validation: When adding a new child or processing payments, the application validates input data for correct types and value ranges. For instance, the child's age must be a positive integer, and payment amounts must also be positive numbers. The application uses try-except blocks to catch any type conversion errors, displaying an informative messagebox to the user in case of invalid input.

4.2 Operational Constraints

Attendance Limitation: The application enforces a daily attendance limit to ensure compliance with capacity regulations. This is managed by counting the number of children scheduled for a particular date and comparing it against the predefined limit. If the limit is reached, an error message is presented, and the user is prevented from adding more children to that date's attendance.

Finalized Month Modification: To prevent retroactive changes that could lead to inconsistencies in attendance records and billing, the application restricts modifications to the attendance data for months that have been finalized. Attempting to edit attendance for a finalized month triggers an error message, informing the user that no further changes are allowed.

4.3 User Interface Feedback

Message Boxes for Immediate Feedback: Throughout the application, tkinter's messagebox module is extensively used to provide immediate, contextual feedback to the user in response to actions. Whether it's to confirm successful operations, warn of potential issues, or alert to errors, message boxes ensure that users are well-informed of the application's state and their actions' outcomes.

Consistent Error Messaging: To maintain clarity and consistency in user communication, the application adopts a uniform approach to error messaging. This includes clear, concise descriptions of the issue encountered and, where appropriate, suggestions for resolving the error. This consistency helps users quickly understand and rectify issues, improving the overall user experience.

These error handling measures are integral to the application's design, ensuring that users can navigate and utilize its features with confidence. By anticipating potential errors and providing clear, informative feedback, the application not only guards against data inconsistencies and operational errors but also enhances user satisfaction and trust in the system.

5.0 Conclusion

In conclusion, our project journey from conceptualizing a basic command-line interface to deploying a fully functional GUI-based daycare management system has been both challenging and rewarding. Through the integration of Python's versatile libraries like Tkinter and tkcalendar, we have created an application that not only meets the needs of an at-home daycare but also demonstrates the powerful capabilities of Python for real-world problem-solving. Each step, from managing child data to processing payments and scheduling, deepened our practical understanding of Python and reinforced the importance of user-centric design in software development. In our project's culmination, we acknowledge the instrumental role of tools like GitHub Copilot, which greatly accelerated our learning curve with Tkinter for GUI development. Embracing these resources, we were able to translate our foundational Python knowledge into a tangible, user-friendly interface that addresses the practical needs of daycare management. This process not only enhanced our technical proficiency but also enriched our understanding of how integrated development environments and collaborative tools can streamline the creation of sophisticated software solutions. Our journey has been a profound testament to the power of leveraging cutting-edge tools to turn concepts into reality in the realm of programming.