Today I will be presenting my Python application titled **“Star Steps.”**

The goal behind **Star Steps** was to develop a python based application with the motivation to support all children while remaining sensitive to neurodiverse needs, highlighting how coding applications have the opportunity to promote inclusivity, routines, and empowerment through everyday steps.

Upon launching the application, users are greeted with a **minimalist and engaging home page**. The interface was deliberately designed to be visually calming, utilising a soft background colour (#C8E6ED) to reduce overstimulation, alongside thoughtfully styled buttons to encourage interaction. The home page also displays the **current date and time**, which enhances both aesthetic appeal and functionality. A clearly labelled **“Start The Day”** button at the bottom of the page allows users to seamlessly begin their daily routine.When users proceed to the **timer page**, the application presents a structured sequence of daily tasks, including *“Wake up and make your bed,” “Brush your teeth,”* and *“Have breakfast.”* Each task is paired with a **countdown timer**, visually guiding the child through the step and helping maintain focus. Task durations are tailored to the activity, shorter tasks such as brushing teeth are allocated brief timers, while longer tasks, such as attending school and reading, are given extended durations.

The application leverages several **core Python modules and libraries** to achieve both functionality and visual appeal.

→ **Tkinter** is utilised to create the graphical user interface, including the Canvas for rendering text, images, and custom buttons, as well as message boxes for step specific notifications.

→ **Datetime** dynamically displays the current date and time across the interface.

→ **Pillow (PIL)** handles image loading and resizing, enabling the integration of external designs on canvas.

→ **OS** ensures that required images exist before attempting to load them, preventing runtime errors.

→ **Time** supports the countdown logic that drives the step-by-step timer functionality.

The **StarStepsApp class** encapsulates the application’s core logic, maintaining the **current page, current step, and timer state**. Methods for page rendering, timer control, and navigation are grouped within the class.

The application window is fixed at **450x900 pixels**, emulating a mobile device layout, which maintains consistent positioning and visual structure across all pages. The **canvas** acts as the primary interface host, rendering text, shapes, images, and custom rounded buttons with precise placement and visual consistency.

The load_background() method dynamically loads and resizes images for both routine and motivational pages. The use of os.path.exists() ensures that missing files do not crash the application, defaulting to coloured rectangles where necessary, maintaining interface consistency. The home page combines date and time display with a central **“Start The**

**Day”** button, providing an intuitive entry point into the routine. Rounded buttons are implemented with configurable size, shape, and colour, enhancing both usability and aesthetic appeal.

The timer page is central to the application, prominently displaying each task and its allocated duration. Timers are formatted consistently, supporting both minutes and hours for longer tasks, ensuring clarity. The **timer controls**—implemented through toggle_timer(), reset_timer(), and run_timer()—leverage Tkinter’s after() function to update every second without freezing the interface. Start, stop, and reset buttons allow users to manage each task.

Navigation between tasks is managed by the go_to_next_step() function. After completing step six, corresponding to school, the application transitions to *“You completed your morning routine.”* Then transitioning to the evening routine, ultimately returning to the home page after completion.

Interactive features, such as on_button_hover() and on_button_leave(), enhance usability by providing immediate visual feedback when hovering over clickable elements. These subtle but important details are critical for maintaining engagement, particularly for children with attention challenges.

Displaying live date and time required careful formatting using Python’s strftime() method. The application converts datetime objects into a human readable format: %B for the full month, %d for the day, %Y for the year, %I for 12 hour format, %M for minutes, %S for seconds, and %p for AM/PM. These formatted strings are continuously updated in the interface using Tkinter’s itemconfig() method, ensuring a dynamic display that does not interrupt the timer or other interface functions.

The main() function serves as the **entry point** of the application, creating the Tkinter window, instantiating the StarStepsApp class, and starting the event loop with root.mainloop(). This loop ensures continuous responsiveness, handling user input and dynamic updates in real time.

In conclusion, this assessment has deepened my understanding of how coding drives application functionality and technological innovation. Developing *Star Steps* extended my technical skills while fostering critical insight into the relationship between design thinking, user experience, and problem-solving. Overcoming challenges such as visual integration, time-based features, and platform constraints strengthened my adaptability and reflective practice. I now approach technology not merely as a user, but as a purposeful designer capable of creating and evaluating digital tools that respond effectively to diverse user needs.