**Appendix**

**Application Overview-**

The application being developed is a data visualization or tracking tool, potentially for monitoring global responses to the COVID-19 pandemic. The database includes data on lockdowns or significant restrictive measures implemented by countries and regions, along with dates and relevant links to news sources or official announcements. Here is an in-depth analysis of the code script submitted as well as how it implements essential features.

**Core Functionalities of the Code-**

**1. Loading Images for Theme Elements**

The script starts by defining a procedure LoadImages which loads images from a specified directory. These images are essential for styling various GUI elements like buttons, scrollbars, and other components. The images are stored in a dictionary I, allowing easy access when configuring styles for widgets.

**2. Theme Creation and Configuration**

Using ttk::style theme create, the script creates a new theme derived from the default theme. It sets global style properties that apply across all widgets unless overridden. This includes setting background colors, foreground colors, fonts, etc.

ttk::style theme create forest-dark -parent default -settings {

# Global style properties

ttk::style configure . -background $colors(-bg) ...

ttk::style map . -foreground [list disabled $colors(-disabledfg)]

...

}

**3. Widget-Specific Styles**

The script meticulously configures each type of widget (TButton, TCheckbutton, TCombobox, etc.):

Button Styles: It configures padding, width, and other properties, and uses images for different states (normal, hover, pressed).

ttk::style configure TButton -padding {8 4 8 4} -width -10 -anchor center

ttk::style element create Button.button image [list $I(rect-basic) ...] -border 4 -sticky nsew

Combobox and Spinbox: Similar to buttons, but with additional elements like dropdown arrows and input fields styled differently.

ttk::style element create Combobox.field image ...

ttk::style element create Spinbox.field image ...

**4. Dynamic Layouts**

For each widget type, the script specifies how children elements (like labels and icons) should be arranged using ttk::style layout.

ttk::style layout TButton {

Button.button -children {

Button.padding -children {

Button.label -side left -expand true

}

}

}

**5. Integration with Tkinter**

While the script sets up the theme in Tcl/Tk syntax, it would be used within a Python Tkinter application where widgets are instantiated using these styles. For example:

import tkinter as tk

from tkinter import ttk

root = tk.Tk()

ttk.Style().theme\_use('forest-dark')

button = ttk.Button(root, text="Click Me")

button.pack()

root.mainloop()

**How does the application works-**

How the Code Works:

1. Theme Definition: The code begins by establishing a new ttk theme named either forest-dark or forest-light. These themes inherit from the default theme but override certain aesthetic properties like colors, button images, and other widget styles.
2. Image Loading: It includes a procedure to load images from a specific directory. These images are then used to style various elements of the GUI, such as buttons, checkboxes, and scrollbars.
3. Style Settings: The script sets up detailed configurations for different GUI components. For each type of widget (TButton, TCombobox, etc.), it specifies how the widget should look under various conditions (e.g., when hovered over, clicked, or disabled).
4. Layout Configuration: For each widget, the script defines a layout which controls how the elements of a widget (like labels inside a button) are organized and displayed.

**Types of User Interactions as Input:**

1. Widget Selection: Users can interact with various widgets like buttons, checkboxes, comboboxes, etc., that you've styled using your custom theme. The input here involves selecting these elements with mouse clicks or keyboard navigation.
2. Data Entry: For widgets like text entries, spinboxes, or date pickers, the user can input data directly into the fields.
3. Options and Commands: In cases where widgets like buttons trigger commands, the user’s click is the input that invokes these commands. For instance, pressing a 'Submit' button styled by your theme would typically execute a function attached to that button.

**Running the Code on Your Device:**

To use this application, follow these steps-

Ensure Python and Tkinter are Installed:

You can check this by running python -m tkinter from your command line or terminal. A small window should pop up if Tkinter is correctly installed.

Convert Tcl Theme Definitions to Python:

If the theme is defined using Tcl (like your scripts), you usually need these definitions loaded into your Tkinter application. Python's Tkinter can execute Tcl commands with the tk.call() method.

Setting Up the Python Script:

Write a Python script that launches a Tkinter window and applies the theme.

Run the Script:

Save your Python script with a .py extension and run it from your command line using python path/to/your\_script.py.

References-

<https://www.thestatesman.com/world/afghan-govt-imposes-lockdown-coronavirus-cases-increase-15-1502870945.html>

<https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Albania>

<https://www.garda.com/crisis24/news-alerts/325896/algeria-government-implements-lockdown-and-curfew-in-blida-and-algiers-march-23-update-7>

<https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Andorra>

<https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Angola>

<https://www.google.ca/>