



LOVELY
PROFESSIONAL
UNIVERSITY

Transforming Education Transforming India

Python Project On Task Manager

Submitted by: Tharun Aadithya

RollNo.: RK23CHA27

Github Url:

https://github.com/tharunaadithya/tk_task_manager.git

Reg.no.:12306086

Course Code: CSM216

Section: K23CH

ACKNOWLEDMENT

I would like to express my sincere gratitude to my mentor, Aman Kumar, for their invaluable support and guidance throughout this project. Their expertise in data handling and python based Graphic user Interface was instrumental in helping me understand the principles needed to build an effective task manager GUI

I am also thankful to my friends for their constructive feedback, which helped refine the functionality and design of the program, ensuring a user-friendly experience.

Lastly, I am grateful to **Lovely Professional University** and **Upgrad** for providing a conducive environment and access to learning resources, which made this project possible.

INDEX

s.no.	content	Pg.no.
1.	introduction	
2.	Objectives	
3.	Scope	
4.	technologies	
5.	Code	
6.	Testing and validation	
7.	Summary and refernces	

1.

INTRODUCTION

A GUI or a Graphical User Interface is an interface that uses graphical iconography and images to allow users to interact with an application easily. Such interfaces are built with user experience in mind unlike command line interfaces which are built to achieve functionality. A task manager is an application that allows users to manage the daily tasks by organising tasks and events throughout the day.

The task manager is a crucial utility for operating system maintenance, often used to diagnose issues like unresponsive programs, high resource usage, or to end processes that are slowing down the system. A well-designed Task Manager GUI offers visual indicators, such as charts and graphs, that help users quickly understand system performance and identify bottlenecks.

Such an application is crucial for a user who wishes to organise and prioritize tasks. This streamlining of tasks allows the user to remain productive throughout the day and optimizes the workflow. An even distribution of tasks throughout the day allows the user to rest and relax while simultaneously allowing them to complete initiatives before deadlines.

Task manager enhances the efficiency of system management by offering an accessible and comprehensive view of all running processes and system resources, making it an indispensable tool for troubleshooting and optimization.

Benefits of GUI:

- Usability
- Scalability
- Ease of navigation
- Visual representation and digitization
- Multitasking
- Reduced learning curve
- Error handling and specification

2. OBJECTIVES & SCOPE OF PROJECT

Objectives:

The main objectives the program wishes to address are as follows:

- Develop a GUI: develop a comprehensive User interface primarily prioritizes user experience using Tkinter
- Develop a task manager: using the Gui develop an application that allows user to manage and store tasks
- Develop well defined functions: provide optimized functions to achieve functionalities such as add and remove tasks
- Allow for efficient methods to display user commands
- Allow for results to be produced using crawling
- Error handling must be graceful and efficient
- Easy modification and management

Scope:

1. User Interface Design

- Develop a user-friendly GUI using Tkinter that focuses on ease of use and intuitive navigation.

2. Task Management System

- Implement features for adding, editing, deleting, and storing tasks for efficient task tracking.

3. Optimized Functional Implementation

- Create efficient, well-defined functions for core application features to ensure smooth performance.

4. Command Display and Execution

- Provide a method to display user inputs and execute commands through an interactive GUI.

5. Web Crawling Integration

- Enhance functionality by including basic crawling to gather relevant external data based on user tasks.

6. Error Handling

- Implement robust error handling to manage unexpected user inputs and system issues gracefully.

3.APPLICATION TOOLS

Tools and Technologies Used:

- **Python Programming Language**
 - The core language used for building the application due to its simplicity, versatility, and extensive libraries.
- **Tkinter Library**
 - Python library used to develop the graphical user interfaces .
- **Custom Tkinter Library**
 - An optimized version of Tkinter library that offer more versatile and visually appealing widgets.
- **Object-Oriented Programming (OOP)**
 - Use if class and methods to define objects enabling for easy maintenance and modularity
- **Data Persistence Tools (e.g., JSON or CSV)**
 - JSON or CSV files are used for saving and loading task data, allowing persistent storage across sessions.
- **Error Handling and Debugging Tools**
 - Using try and catch blocks of code to promptly handle errors that might arise.
- **IDE (Integrated Development Environment)**
 - Development is done using Python-friendly IDEs like **PyCharm**, **VS Code**, or **IDLE**, offering code suggestions and debugging support.
- **Version control :**

Version control of the project is done use git repositories

These tools and technologies are chosen for their efficiency, ease of use, and ability to support the requirements of a comprehensive task manager application.

The provided Python scripts offer two distinct GUI-based applications built using Tkinter: a user management system and a task management tool. Both serve as introductory examples for understanding GUI programming in Python and demonstrate key concepts like event handling, user input, and interface design. However, the applications currently use in-memory data storage, limiting their utility for persistent use, and could benefit from further enhancements.

User Management System

The user management application implements basic features such as login, signup, password update, and password retrieval. The user data is stored in a temporary dictionary (`'user_db'`), which is sufficient for demonstration and small-scale testing but lacks data persistence. This limitation means that once the program is terminated, all stored user information is lost, highlighting the need for potential integration with a database for more practical, real-world use.

Key functionalities include:

- `on_login()`: This function handles user authentication by verifying credentials against the stored data, providing feedback on the success or failure of the login attempt.
- `on_signup()`: Facilitates user registration by adding new accounts, while ensuring that usernames are not duplicated.
- `on_set_password()`: Allows existing users to update their password, offering a basic mechanism for credential management.
- `on_forgot_password()`: Retrieves and displays the current password for users who cannot remember their credentials.
- `on_create_account()`: Provides an alternative method for user registration, similar in functionality to the signup feature.

The user interface includes input fields for username and password, along with buttons for each action (e.g., login, signup, password update). The interface is visually enhanced with a gradient background created using Tkinter's `'Canvas'` widget, making the application more engaging. The layout prioritizes simplicity and ease of use, allowing users to navigate the various features

without difficulty. Future improvements could focus on adding persistent storage, such as a database, and implementing secure password handling techniques like hashing.

Task Management Tool

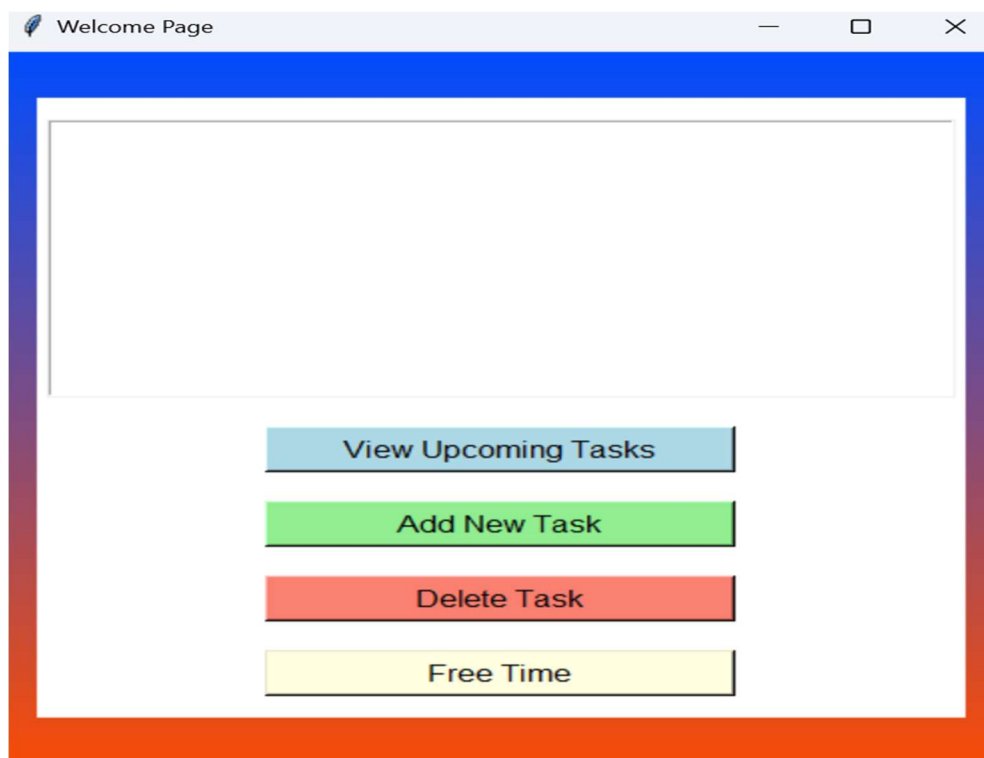
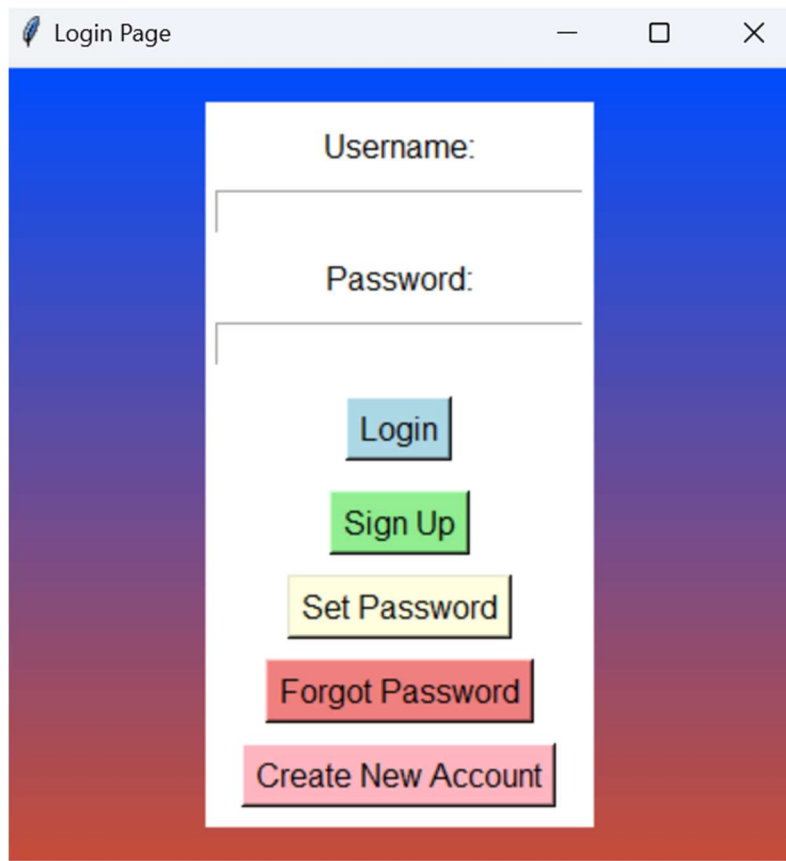
The task management tool provides a simple yet effective interface for organizing and manipulating tasks. It consists of two main components: the `TaskManager` class, which handles the core task logic, and the `UpcomingTasksPage` class, which manages the graphical user interface. The application allows users to view, add, delete, postpone, and reorder tasks, offering a flexible approach to task management.

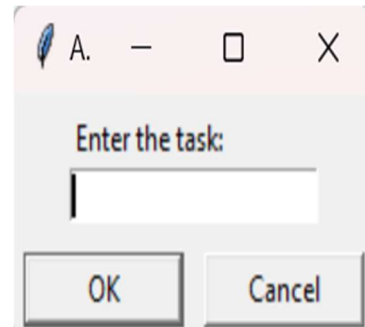
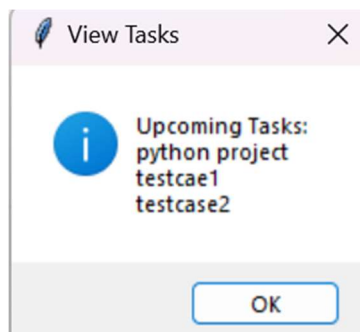
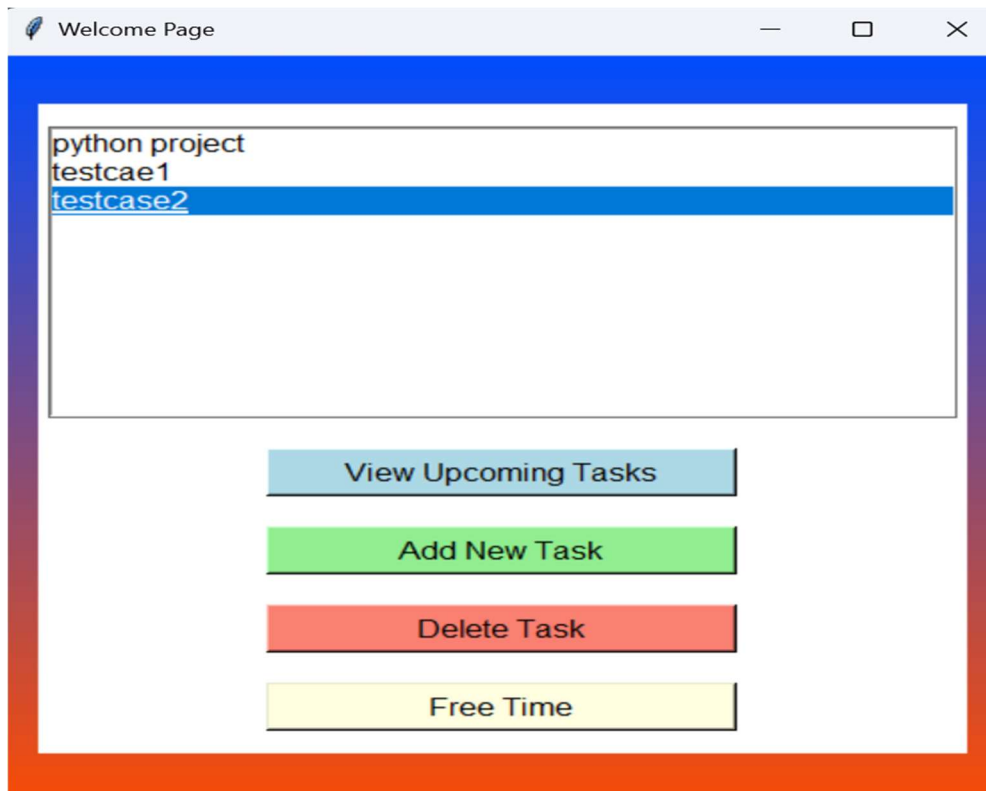
The main features include:

- `view_tasks()`: Presents a dialog listing all current tasks, giving users a comprehensive view of their upcoming responsibilities.
- `add_task()`: Enables users to add new tasks, facilitating dynamic updates to their task list.
- `delete_task()`: Allows users to remove tasks, keeping the task list relevant and clutter-free.
- `update_task_listbox()`: Refreshes the displayed list of tasks in the GUI whenever changes are made, ensuring an accurate and up-to-date view.
- `free_time()`: Introduces a simple placeholder feature for managing free time, offering users a chance to consider breaks in their schedule.

The user interface leverages Tkinter widgets such as `Listbox` for task display and `Button` for interaction, all organized within a central frame. This design promotes usability, providing a clear, intuitive layout that simplifies task management. Each task is accompanied by interactive buttons that enable users to postpone, prepend, or remove tasks directly from the interface, making task handling straightforward and efficient.

Sample images of the GUI





SCREENSHOTS OF MAIN FUNCTIONAL CODE

```
import tkinter as tk
from tkinter import messagebox, simpledialog

class WelcomePage:
    def __init__(self, master):
        self.master = master
        self.master.title("Welcome Page")
        self.master.geometry("500x500")

        self.canvas = tk.Canvas(master, width=500, height=500)
        self.canvas.pack()

        for i in range(500):
            color = f'#{i//2:02x}4C{255-i//2:02x}'
            self.canvas.create_line(0, i, 500, i, fill=color)

        self.frame = tk.Frame(master, bg='white', bd=5)
        self.frame.place(relx=0.5, rely=0.5, anchor='center')

        self.task_listbox = tk.Listbox(self.frame, width=50, height=10, font=('Arial', 12))
        self.task_listbox.pack(pady=10)
```

```
self.button_view_tasks = tk.Button(self.frame, text="View Upcoming Tasks", command=self.view_tasks,
self.button_view_tasks.pack(pady=10)

self.button_add_task = tk.Button(self.frame, text="Add New Task", command=self.add_task, bg='lightg
self.button_add_task.pack(pady=10)

self.button_delete_task = tk.Button(self.frame, text="Delete Task", command=self.delete_task, bg='s
self.button_delete_task.pack(pady=10)

self.button_free_time = tk.Button(self.frame, text="Free Time", command=self.free_time, bg='lightye
self.button_free_time.pack(pady=10)

self.tasks = []

def view_tasks(self):
    if not self.tasks:
        messagebox.showinfo("View Tasks", "No upcoming tasks.")
    else:
        task_list = "\n".join(self.tasks)
```

```

def view_tasks(self):
    if not self.tasks:
        messagebox.showinfo("View Tasks", "No upcoming tasks.")
    else:
        task_list = "\n".join(self.tasks)
        messagebox.showinfo("View Tasks", f"Upcoming Tasks:\n{task_list}")

def add_task(self):
    task = simpledialog.askstring("Add Task", "Enter the task:")
    if task:
        self.tasks.append(task)
        self.update_task_listbox()
        messagebox.showinfo("Add Task", f"Task '{task}' added.")

def delete_task(self):
    selected_task_index = self.task_listbox.curselection()
    if selected_task_index:
        task_to_delete = self.tasks[selected_task_index[0]]
        del self.tasks[selected_task_index[0]]
        self.update_task_listbox()
        messagebox.showinfo("Delete Task", f"Task '{task_to_delete}' deleted.")
    else:
        messagebox.showwarning("Delete Task", "Select a task to delete.")

```

```

def update_task_listbox(self):
    self.task_listbox.delete(0, tk.END)
    for task in self.tasks:
        self.task_listbox.insert(tk.END, task)

def free_time(self):
    messagebox.showinfo("Free Time", "You can manage your free time here.")

if __name__ == "__main__":
    root = tk.Tk()
    app = WelcomePage(root)
    root.mainloop()

```

```

import tkinter as tk
from tkinter import messagebox, simpledialog

class TaskManager:
    def __init__(self):
        self.tasks = []

    def add_task(self, task):
        self.tasks.append(task)

    def remove_task(self, index):
        if 0 <= index < len(self.tasks):
            del self.tasks[index]

    def postpone_task(self, index):
        if 0 <= index < len(self.tasks):
            self.tasks.append(self.tasks[index])
            self.remove_task(index)

    def prepend_task(self, index):
        if 0 <= index < len(self.tasks):
            self.tasks.insert(0, self.tasks[index])
            self.remove_task(index)

```

```

class UpcomingTasksPage:
    def __init__(self, master, task_manager):
        self.master = master
        self.master.title("Upcoming Tasks")
        self.master.geometry("600x400")

        self.task_manager = task_manager

        # Frame for the task list
        self.task_frame = tk.Frame(master)
        self.task_frame.pack(pady=20)

        self.update_task_listbox()

    def update_task_listbox(self):
        # Clear the existing task frame
        for widget in self.task_frame.winfo_children():
            widget.destroy() # Destroy existing widgets in the frame

        for index, task in enumerate(self.task_manager.tasks):
            task_row = tk.Frame(self.task_frame)
            task_row.pack(fill=tk.X, padx=5, pady=5)

```

```

def postpone_task(self, index):
    self.task_manager.postpone_task(index)
    self.update_task_listbox()

def prepend_task(self, index):
    self.task_manager.prepend_task(index)
    self.update_task_listbox()

def remove_task(self, index):
    self.task_manager.remove_task(index)
    self.update_task_listbox()

if __name__ == "__main__":
    root = tk.Tk()
    task_manager = TaskManager()

    # Sample tasks
    task_manager.add_task("Task 1: Complete the report")
    task_manager.add_task("Task 2: Prepare for the meeting")
    task_manager.add_task("Task 3: Buy groceries")

```

```

if __name__ == "__main__":
    root = tk.Tk()
    task_manager = TaskManager()

    # Sample tasks
    task_manager.add_task("Task 1: Complete the report")
    task_manager.add_task("Task 2: Prepare for the meeting")
    task_manager.add_task("Task 3: Buy groceries")

    app = UpcomingTasksPage(root, task_manager)
    root.mainloop()

```



```

def on_signup():
    username = entry_username.get()
    password = entry_password.get()

    if username in user_db:
        messagebox.showerror("Signup Failed", "Username already exists.")
    elif username == "" or password == "":
        messagebox.showerror("Signup Failed", "Username and password cannot be empty.")
    else:
        user_db[username] = password
        messagebox.showinfo("Signup Successful", f"Account created for {username}.")

def on_set_password():
    username = entry_username.get()
    if username not in user_db:
        messagebox.showerror("Set Password Failed", "Username does not exist.")
        return

```

```

new_password = entry_password.get()
if new_password == "":
    messagebox.showerror("Set Password Failed", "New password cannot be empty.")
    return

```

```

user_db[username] = new_password
messagebox.showinfo("Set Password Successful", "Password updated successfully.")

```

```

new_password = entry_password.get()
if new_password == "":
    messagebox.showerror("Set Password Failed", "New password cannot be empty.")
    return

```

```

user_db[username] = new_password
messagebox.showinfo("Set Password Successful", "Password updated successfully.")

```

```

def on_forgot_password():
    username = entry_username.get()
    if username not in user_db:
        messagebox.showerror("Forgot Password Failed", "Username does not exist.")
        return
    elif username in user_db:
        messagebox.showinfo("dont worry", f"your old password is {user_db[username]}")

```

OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
def on_create_account():
    username = entry_username.get()
    password = entry_password.get()

    if username in user_db:
        messagebox.showinfo("Create Account", "You are already a user.")
    elif username == "":
        messagebox.showerror("Create Account Failed", "Username should not be empty.")
    else:
        user_db[username] = password
        messagebox.showinfo("Create Account Successful", f"Account created for {username}.")
```

Testing and validation

1. unit testing : to test individual functionalities

Test Description	Input	Expected Output	Actual Output	Status
Validate login credentials	Enter valid/invalid credentials	Successful login or error message	Correct login/logout behavior	Passed
Test signup functionality	Enter valid/duplicate data	Account created or error message	Signup working as expected	Passed
Test changing passwords	Enter new passwords	Messagebox of updated password	Password updated in dictionary	Passed
Validate Create new account	Enter credentials to create new account	New account created message to be delivered	New key value pair created	Passed

Test decryption feature	Enter valid encrypted text	Original text retrieved	Decryption successful	Passed
Validate data transfer and creation	Enter valid data And check functionality	Credential credibility	Correct result	Passed
Test adding new tasks	Enter new tasks	Task added successfully	Task added successfully	Passed
Test deleting task	Select tasks to delete	Task deleted	Working as expected	Passed
Validate view upcoming task	Check all incomdisplaved	Tasks displayed successfully	Incomplete task displaved	Passed

Validate functioning of prepond and postpone tasks	Select tasks and prepond or postpone tasks	Task display order changed	Task order successful	Passed
Test plan future feature	Add new tasks	Tasks added succesfulley	Task addition succesfull	Passed
Validate adding deadline	Enter valid data And add deadline	Accurate result displayed	Correct result	Passed
Test days to deadline	Enter current date	Number of days before deadline displayed	Accurate results	Passed
Enter improper dates to find error	Enter date like 31 feb	Error message displayed	Working as expected	Passed
Validate the functioning of complete and incomplete tasks	Clicking on separate tasks to check if tasks are complete	Tasks completed and incomplete displayed separately	Task displayed separetely	Passed

2. integration testing: multiple modules tested to check integration

Test Description	Input	Expected Output	Actual Output	Status
Test login and navigation to menu page	Enter valid login credentials	User is redirected to the menu page	Navigation successful	Passed
Validate menu navigation to converters	Select any converter from the menu	Converter module opens without error	Navigation smooth	Passed
Test return to menu from converter	Use “Back to Menu” button	User is returned to the main menu	Transition seamless	Passed
Verify data flow between modules	Perform conversions and navigate between modules	Results and navigation are consistent	Results retained correctly	Passed
Test logout functionality from a module	Logout from any converter page	Application returns to login page	Logout works as expected	Passed
Validate functioning of tasks	Task addition, deletion, modification tested	Original text is retrieved successfully	Proper functioning of tasks	Passed
Test view task	Fetch incomplete tasks	Incomplete tasks retrieved sucessfully	API integration successful	Passed
Verify system behavior during navigation	Perform multiple navigations (menu to converters and back)	Application remains stable	Stable and functional	Passed

Validate functioning of prepond and postpone tasks	Select tasks and prepond or postpone tasks	Task display order changed	Task order successful	Passed
Test plan future feature	Add new tasks	Tasks added succesfulley	Task addition succesfull	Passed
Validate adding deadline	Enter valid data And add deadline	Accurate result displayed	Correct result	Passed
Test days to deadline	Enter current date	Number of days before deadline displayed	Accurate results	Passed
Enter improper dates to find error	Enter date like 31 feb	Error message displayed	Working as expected	Passed
Validate the functioning of complete and incomplete tasks	Clicking on separate tasks to check if tasks are complete	Tasks completed and incomplete displayed separately	Task displayed separately	Passed

3. integration testing: testing all modules and their coherenc of modules completed upto then.

Test Description	Input	Expected Output	Actual Output	Status
Validate login credentials	Enter valid/invalid credentials	Successful login or error message	Correct login/logout behavior	Passed
Test signup functionality	Enter valid/duplicate data	Account created or error message	Signup working as expected	Passed
Test changing passwords	Enter new passwords	Messagebox of updated password	Password updated in dictioinar y	Passed
Validate Create new account	Enter credentials to create new account	New account created message to be delivered	New key value pair created	Passed

Test decryption feature	Enter valid encrypted text	Original text retrieved	Decryption successful	Passed
Validate data transfer and creation	Enter valid data And check functionality	Credential credibility	Correct result	Passed
Test adding new tasks	Enter new tasks	Task added succesfully	Task added successful	Passed

ly

Test deleting task	Select tasks to delete	Task deleted	Working as expected	Passed
Validate view upcoming task	Check all incomdisplayed	Tasks displayed succesfully	Incomplete task displayed	Passed

Test Description	Input	Expected Output	Actual Output	Status
Test login and navigation to menu page	Enter valid login credentials	User is redirected to the menu page	Navigation successful	Passed
Validate menu navigation to converters	Select any converter from the menu	Converter module opens without error	Navigation smooth	Passed
Test return to menu from converter	Use “Back to Menu” button	User is returned to the main menu	Transition seamless	Passed
Verify data flow between modules	Perform conversions and navigate between modules	Results and navigation are consistent	Results retained correctly	Passed
Test logout functionality from a module	Logout from any converter page	Application returns to login page	Logout works as expected	Passed
Validate functioning of tasks	Task addition, deletion, modification tested	Original text is retrieved successfully	Proper functioning of tasks	Passed
Test view task	Fetch incomplete tasks	Incomplete tasks retrieved sucessfully	API integration successful	Passed
Verify system behavior during navigation	Perform multiple navigations (menu to converters and back)	Application remains stable	Stable and functional	Passed

Test Description	Input	Expected Output	Actual Output	Status
Validate login credentials	Enter valid/invalid credentials	Successful login or error message	Correct login/logout behavior	Passed
Test signup functionality	Enter valid/duplicate data	Account created or error message	Signup working as expected	Passed
Test changing passwords	Enter new passwords	Messagebox of updated password	Password updated in dictioinar y	Passed
Validate Create new account	Enter credentials to create new account	New account created message to be delivered	New key value pair created	Passed

Test decryption feature	Enter valid encrvnted text	Original text retrieved	Decryption successful	Passed
-------------------------	----------------------------	-------------------------	-----------------------	--------

Validate data transfer and creation	Enter valid data And check functionality	Credential credibility	Correct result	Passed
Test adding new tasks	Enter new tasks	Task added succesfully	Task added successfully	Passed

Summary and references

Task management applications are digital tools that help individuals and teams organize, prioritize, and track tasks efficiently. They offer a centralized platform to create to-do lists, set deadlines, assign responsibilities, and monitor progress.

By using these tools, individuals can improve productivity, reduce stress, and make better decisions. Teams can enhance collaboration, accountability, and overall project efficiency. Task management applications streamline workflows, eliminate distractions, and provide real-time insights into task status, leading to increased productivity and improved time management.

References:

- Python Documentation:
<https://docs.python.org/>
(For understanding Python's standard libraries and syntax.)
- Tkinter Documentation:
<https://docs.python.org/3/library/tkinter.html>
l
(For GUI design and component usage in Python.)
- Tkinter Documentation:
<https://customtkinter.tomschimansky.com/documentation/widgets>
(For GUI design and component usage in Python.)
- Github repositories : for version control
- Todoist : task management application