A

Mini Project Report

On

“ MODERN GUI - WINDOWS SOFTWARE APPLICATION ”

Submitted By

Name	Enrollment No
Vivek Shashikant Shende	(2023011032086)
Vaibhav Santosh Ukande	(2023011032096)
Yash Jayavant Pawar	(2023011032101)
Yash Vishvanath Hasabe	(2023011032024)
Vishwajeet Mohan Wagh	(2023011032098)

Under the Guidance of

Mr. Sandip Rabade

Department of Computer Science & Engineering

School of Engineering & Technology

D. Y. Patil Agriculture & Technical University, Talsande

Academic Year

2024-25

D. Y. Patil Agriculture & Technical University, Talsande

Department of Computer Science & Engineering

Certificate

This is to certify that the project work entitled

"MODERN GUI - WINDOWS SOFTWARE APPLICATION"

Submitted by

Name	Roll No/PRN
Vivek Shashikant Shende	(2023011032086)
Vaibhav Santosh Ukande	(2023011032096)
Yash Jayavant Pawar	(2023011032101)
Yash Vishvanath Hasabe	(2023011032024)
Vishwajeet Mohan Wagh	(2023011032098)

In partial fulfillment of requirement for the Semester-I of Third Year in Computer Science & Engineering. This is a record of their work carried out by them under supervision and guidance during academic year 2024-25.

Place:

Date:

Mr. Sandip Rabade	Mr. S. A. Kumbhar	Dr. Jaydeep. B. Patil
Name of Guide	**Project coordinator**	**HOD CSE**

External Examiner

ACKNOWLEDGMENT

We take pleasure in presenting our work done for the project entitled as "MODERN GUI - WINDOWS SOFTWARE APPLICATION "

We express our sincere thanks to **Mr. Sandip Rabade** whose supervision, inspiration and valuable guidance helped us a lot to complete the project. His guidance proved to be the most valuable to overcome all the hurdles in the fulfillment of this project work.

We are very much thankful to Mr. S. A. Kumbhar, Project coordinator and Dr. S. T. Patil, Associate Dean, School of Engineering and Technology for their kind support and valuable guidance.

We would also express gratitude towards our colleagues and friends for the moral and technical support throughout the duration of project work. Also we are also thankful to all those who have helped us in the completion of the project work.

Vivek Shashikant Shende
Vaibhav Santosh Ukande
Yash Jayavant Pawar
Yash Vishvanath Hasabe
Vishwajeet Mohan Wagh

INDEX

1. Synopsis :

In today’s fast-paced digital landscape, the demand for intuitive and efficient software solutions has never been greater. The modern graphical user interface (GUI) window software application project seeks to address this need by offering a state-of-the-art platform that combines functionality with user-centric design.Our project aims to redefine user interaction by leveraging contemporary design principles and advanced technologies. The application is meticulously crafted to enhance user experience through a sleek and responsive interface, streamlined workflows, and robust features tailored to meet both personal and professional needs.

Key objectives of this project include:

- **Performance and Efficiency:** Ensuring the application is optimized for speed and reliability, providing a smooth experience even with high-demand tasks.
- **Scalability:** Developing a flexible architecture that supports future enhancements and integrations, allowing the application to evolve with emerging trends and technologies.
- **Accessibility:** Incorporating features that cater to diverse user needs, including customizable settings

Hardware and Software Requirement

- **Hardware:** PC / Deskstop / Laptop
- **Software:** Py-charm, Python IDE, tkinter

Performance Requirement

- Immediate synchronization
- Efficient handling of large data
- One-Click access

Other Requirement

- Weather detection for notification

CHAPTER: 2 INTRODUCTION

Introduction

2.1 Overview of the Project

In today's fast-paced digital landscape, the demand for intuitive and efficient software solutions has never been greater. The modern graphical user interface (GUI) window software application project seeks to address this need by offering a state-of-the-art platform that combines functionality with user-centric design.Our project aims to redefine user interaction by leveraging contemporary design principles and advanced technologies. The application is meticulously crafted to enhance user experience through a sleek and responsive interface, streamlined workflows, and robust features tailored to meet both personal and professional needs.

Key objectives of this project include:

• Performance and Efficiency: Ensuring the application is optimized for speed and reliability, providing a smooth experience even with high-demand tasks.
• Scalability: Developing a flexible architecture that supports future enhancements and integrations, allowing the application to evolve with emerging trends and technologies.
• Accessibility: Incorporating features that cater to diverse user needs, including customizable settings

2.2 Problem Statement

In the evolving landscape of software applications, users increasingly demand solutions that are not only functional but also intuitive, efficient, and adaptable to their needs. Despite significant advancements in technology, several key issues persist in current GUI window software applications, presenting opportunities for improvement and innovation.

Key Problems :

1. Complex User Interfaces

*** Problem :** Many existing applications suffer from overly complex and cluttered interfaces, which can overwhelm users and hinder productivity. This complexity often results from a lack of user-centered design principles and inadequate consideration of user workflows.

*** Impact :** Users may experience increased cognitive load, reduced efficiency, and a steeper learning curve, leading to frustration and decreased adoption rates.

2.3 Objectives

For a modern GUI (Graphical User Interface) Windows application project, objectives help ensure that the application meets user needs and industry standards. Here are some key objectives you might consider:

1. User Experience (UX)

• Intuitive Design: Create a user-friendly interface that is easy to navigate and understand.

• Responsive Layout: Ensure the application adjusts smoothly to different screen sizes and resolutions.

• Accessibility: Implement features to support users with disabilities, such as keyboard navigation and screen reader compatibility.

2. Functionality

• Core Features: Define and implement the primary functions and features that the application needs to perform.

• Performance: Optimize the application to run efficiently, with fast load times and minimal resource usage.
• Error Handling: Develop robust error handling to manage and report issues gracefully.

3. Aesthetics

• Modern Design: Use contemporary design principles and elements, such as flat design, consistent color schemes, and high-quality icons.
• Branding: Ensure the GUI aligns with the branding guidelines of the organization or product.

CHAPTER: 3
LITERATURE REVIEW

3 - Literature Review / Existing System

3.1 Existing Systems

In the realm of modern GUI window software applications, several established systems and technologies serve as the foundation for current software solutions. These existing systems provide a baseline from which innovations can be measured and developed. Here, we outline the primary systems and their characteristics:

1.Traditional Desktop Environments :

• **Windows OS:** Microsoft Windows has long been a dominant force in desktop environments, providing a rich set of features and a mature ecosystem of applications. Its GUI is known for its familiarity, extensive support, and a wide range of customization options.

• **macOS:** Apple’s macOS offers a polished and integrated user experience with a strong emphasis on aesthetics and performance. It provides a robust set of tools for application development, including native frameworks like Cocoa.

• **Linux Desktop Environments (e.g., GNOME, KDE):** Linux offers a variety of desktop environments, each with its own strengths. GNOME focuses on simplicity and ease of use, while KDE emphasizes customization and feature richness.

3.2 Advantages of the Proposed System

1. User Interface (UI)

• Navigation: The application should provide a consistent and intuitive navigation system, including menus, toolbars, and context menus.
• Forms and Dialogs: Support various forms and dialogs for user input, including validation and error messages.

• Dynamic Content: Update and display content dynamically based on user interactions.

2. User Authentication and Authorization

• Login/Logout: Users must be able to securely log in and out of the application.
• User Roles: Support different user roles with specific permissions and access levels.
• Password Management: Include functionalities for password reset and management.

3. Search and Filtering

• Search Functionality: Implement search features to find specific data or content quickly.

4. Notifications and Alerts

• User Notifications: Alert users to important events, updates, or errors.
• System Alerts: Provide system-wide notifications for critical issues or status changes.

5. Settings and Preferences

• User Preferences: Allow users to configure application settings according to their preferences.
• Application Settings: Support configuration of application-specific settings, such as language, themes, or behavior.
• Offline Functionality
• Offline Mode: If applicable, support functionality that allows users to work offline and synchronize data once they are back online.

CHAPTER: 4
BASIC SYSTEM & ARCHITECTURE

4.1 Outline of Proposed System

To create a modern GUI-based Windows application using Python, developers typically rely on a combination of Python libraries for user interface design, system integration, and backend logic. Below is an overview of the basic system and architecture for such an application.

1. Overview of Components

A modern GUI Windows application generally consists of the following major components:

User Interface (UI): The graphical elements that the user interacts with (buttons, text fields, menus, etc.).

Backend Logic: The application’s business logic that processes data, communicates with databases, APIs, or performs calculations.

Data Management: The part of the system that handles data storage, retrieval, and possibly synchronization with external systems.

System Integration: Ensuring the application works well with the Windows OS, including file system access, hardware interaction, and other system-level functions.

Application State Management: Managing the state of the application and handling user sessions, settings, or configurations.

Below is a breakdown of the architecture for a modern GUI Windows application built with Python:

2. Architecture of a Modern GUI Windows Application

a. Frontend (GUI Layer)

This is the part of the system where users directly interact with the application. The GUI typically consists of several components such as:

Windows and Controls: The primary window and the various controls (buttons, checkboxes, sliders, etc.) that the user interacts with.
Event Handlers: Python event-driven programming helps manage interactions like button clicks, mouse movements, or keyboard input.
Layouts and Views: Organizing and displaying the controls, sometimes dynamically, depending on the size of the window or user actions.
Python libraries like Tkinter, PyQt, Kivy, and wxPython are used for building the graphical user interface:

Tkinter is Python's standard GUI library that is simple to use and widely available.
PyQt provides more advanced and feature-rich GUI capabilities, integrating well with the Qt framework.
Kivy is ideal for developing cross-platform applications, including mobile apps, and provides a more modern touch-based UI.
wxPython is another toolkit that is close to the native look and feel of the Windows desktop.

b. Backend Logic (Business Logic Layer)

The backend logic is responsible for processing the data and handling the application’s core functionality. It includes:

Data Handling and Processing: This layer interacts with the data sources (databases, files, or APIs). For instance, Python's built-in libraries like SQLite, SQLAlchemy, or external APIs can be used for database communication.
Business Rules: Implements the logic that drives the application's features (e.g., calculations, transformations, etc.).
Asynchronous Processing: For tasks that could block the UI, such as downloading data, performing calculations, or interacting with a database, Python's asyncio or threading libraries may be used.

c. Data Management Layer

This layer is responsible for interacting with databases or external data sources, such as:

File System Interaction: Python's os and shutil libraries allow interaction with the file system for reading, writing, and organizing files.
Database Interaction: For applications that need persistent data storage, a database like SQLite or an external database server (e.g., PostgreSQL, MySQL) can be used. Python's SQLAlchemy or sqlite3 can be employed for database communication.
d. System Integration Layer
This layer handles the interaction of the application with the Windows operating system, providing functionalities like:

Native Integration: Using Python's pywin32 library, developers can access Windows-specific features like working with the registry, accessing COM interfaces, managing processes, and other OS-level tasks.
Notifications: Windows-specific system notifications (e.g., taskbar notifications) can be implemented using libraries like win10toast or pystray.
Hardware Access: For hardware integration, such as interacting with serial ports or USB devices, libraries like PySerial are commonly used.

e. Communication Layer (Optional)
If the application requires external communication, such as calling APIs, sending data to a server, or integrating with cloud services, this layer comes into play:

Networking: Libraries such as requests (for HTTP requests), socket (for lower-level network communication), and asyncio (for asynchronous network tasks) can be employed.
Web APIs: If the application needs to interact with web services, RESTful APIs can be accessed using the requests library for data retrieval and submission.

3. Common Python Libraries and Frameworks for GUI Applications

Here is a list of some key libraries and tools used to build a modern GUI Windows software application in Python:

a. GUI Frameworks:

Tkinter: Simple and lightweight, part of the standard Python library.

PyQt: A comprehensive set of Python bindings for the Qt toolkit, used for building cross-platform applications.

wxPython: Provides a native look and feel on Windows systems.

Kivy: Ideal for developing touch-based UIs and cross-platform applications.

b. Database and Data Management:

SQLite: A lightweight, embedded database.

SQLAlchemy: A powerful ORM for SQL-based databases.

Pandas: For data manipulation and analysis (useful for handling large datasets).

c. Networking and API Communication:

requests: For interacting with HTTP APIs.

socket: Low-level networking for TCP/IP communication.

d. Windows Integration and OS-Level Interaction:

pywin32: Provides access to Windows-specific APIs, such as COM, the registry, and more.

win10toast: A library for showing Windows 10 system notifications.

e. Multi-Threading and Asynchronous Operations:

threading: For running tasks in the background to avoid blocking the GUI.

asyncio: For asynchronous programming to improve performance and responsiveness.

f. Packaging and Deployment:

PyInstaller: A tool for packaging Python applications into standalone executables for Windows.

cx_Freeze: Another popular tool for freezing Python apps into executables.

4. Basic System Workflow

User Interaction (Frontend):

User interacts with buttons, text fields, and other controls in the GUI.
Events are triggered (e.g., button clicks, input changes).
Backend Logic Processing:

Upon receiving an event, the backend processes the data (e.g., fetch data from a database, perform calculations).
Results are sent back to the GUI for display.
Data Management:
Data may be stored or retrieved from a database or local files.
If needed, external communication via APIs or network requests can take place.
System Integration (Optional):

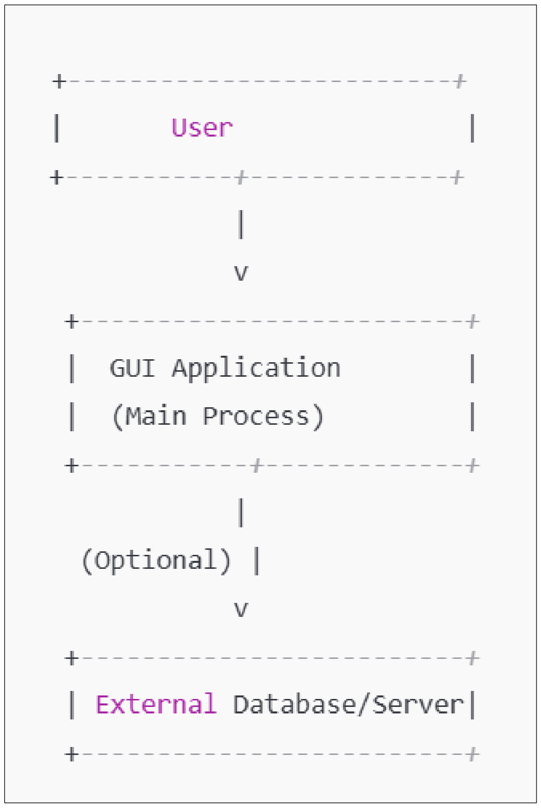
The application may interact with the operating system (e.g., save files to disk, send notifications to the taskbar).
Display of Results:

The results are displayed in the UI (updated controls, new windows, or messages).

CHAPTER: 5 DESIGN

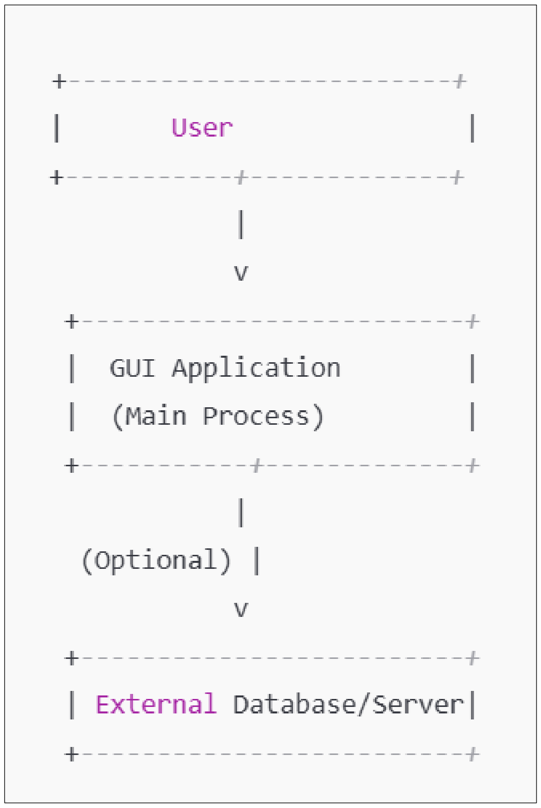
1. Data Flow Diagram

1.1 Level 0 DFD

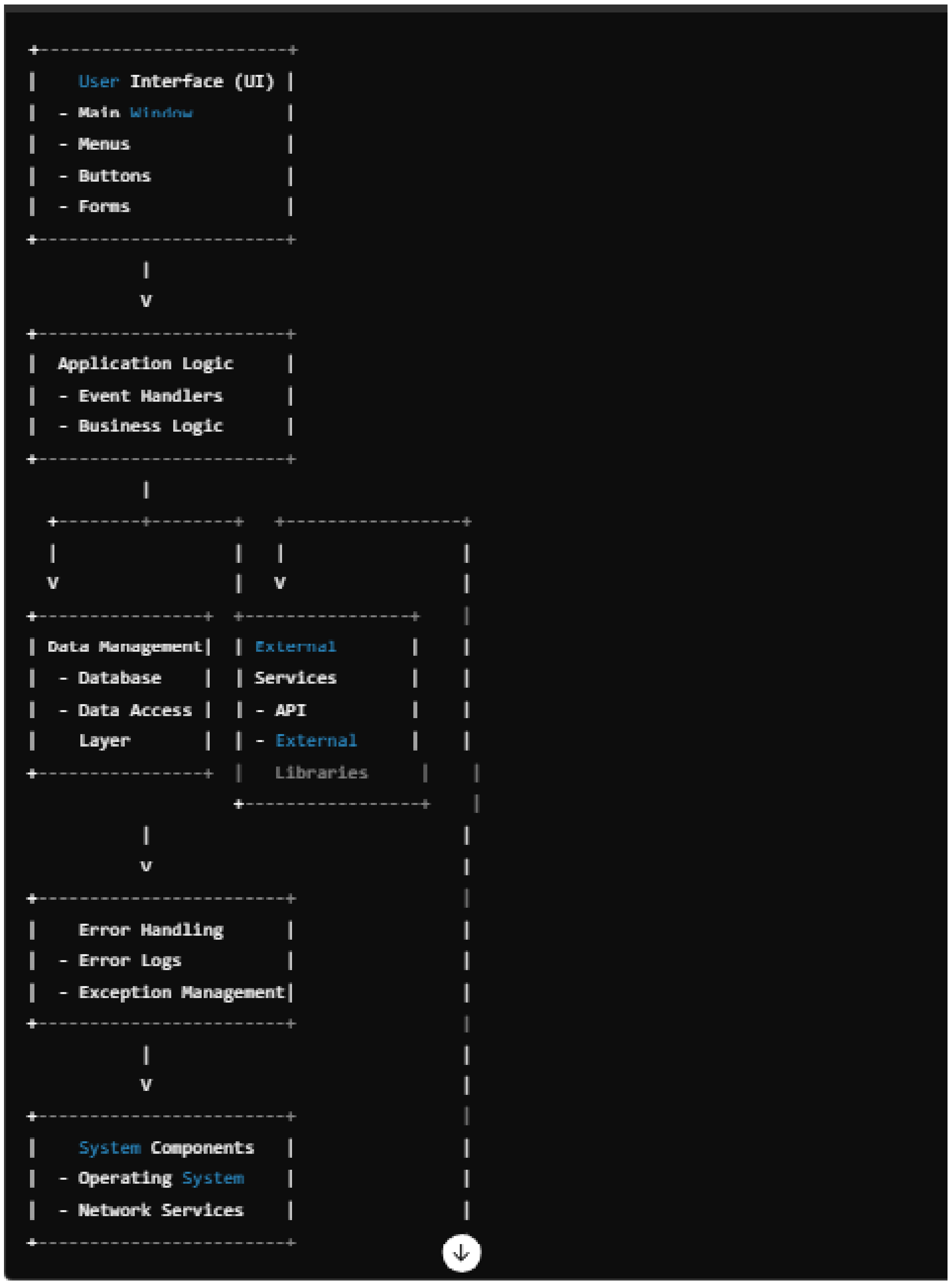


```
+-------------------------+
|       User              |
+-----------+-------------+
            |
            v
 +-------------------------+
 |  GUI Application        |
 |  (Main Process)         |
 +-----------+-------------+
             |
 (Optional) |
            v
 +-------------------------+
 | External Database/Server|
 +-------------------------+
```

1.2 Level 1 DFD



```
+------------------------+
|       User             |
+----------+-------------+
           |
           v
 +-----------------------+
 |  GUI Application      |
 |  (Main Process)       |
 +----------+------------+
           |
  (Optional) |
           v
 +-----------------------+
 | External Database/Server|
 +-----------------------+
```



```
+-----------------------+
|    User Interface (UI) |
|  - Main Window         |
|  - Menus               |
|  - Buttons             |
|  - Forms               |
+-----------------------+
          |
          v
+-----------------------+
| Application Logic     |
|  - Event Handlers     |
|  - Business Logic     |
+-----------------------+
          |
  +--------+--------+   +-----------------+
  |                |   |                 |
  v                |   v                 |
+----------------+ +----------------+    |
| Data Management| | External       |    |
|  - Database    | | Services       |    |
|  - Data Access | | - API          |    |
|    Layer       | | - External     |    |
+----------------+ |   Libraries    |    |
                   +----------------+    |
          |                              |
          v                              |
+-----------------------+                |
|    Error Handling     |                |
|  - Error Logs         |                |
|  - Exception Management|               |
+-----------------------+                |
          |                              |
          v                              |
+-----------------------+                |
|    System Components  |                |
|  - Operating System   |                |
|  - Network Services   |                |
+-----------------------+
```

CHAPTER 6 : IMPLEMENTATION ALGORITHM

6.1 Algorithms

Below is an algorithm for a simple Modern GUI Windows Software Application using Python. The application will demonstrate basic user interactions such as logging in, displaying a main window, submitting and saving data, and exiting the application.

This algorithm assumes the use of a GUI framework such as Tkinter (Python's built-in library for GUI applications). It outlines the key steps for handling user interactions, processing data, and managing the flow of the application.

Algorithm Overview :

- Initialize the Application:

  - Set up the GUI environment.
  - Initialize necessary components (buttons, entry fields, labels, etc.).

- User Login:

  - Display the login form (username and password fields).
  - Validate the credentials (basic validation or mock validation for simplicity).
  - If login is successful, display the main application window.

- Main Application Window:

  - Display buttons, input fields, and data views.
  - Allow the user to submit data, view stored data, or exit the application.

- Submit Data:

- Allow the user to input data (e.g., text, numbers) in a form.
- Validate the input data.
- If valid, save the data to a local storage (e.g., a file or database).

➢ Exit the Application:

Confirm if the user wants to exit.

Close the application window.

Algorithm:

Step 1: Initialize the GUI Application

Initialize GUI window: Set up a window with a title and dimensions.

Add Widgets: Add necessary widgets (e.g., labels, entry fields, buttons).

Username entry field.

Password entry field.

Login button.

Main window button(s) for submitting data, viewing data, and exiting.

Step 2: User Login Process

Display Login Screen: Show username and password entry fields with a "Login" button.

Handle Login Button Click:

Retrieve user input (username and password).

Validate user input.

If credentials are valid, proceed to the main application window.

If credentials are invalid, show an error message.

Step 3: Main Application Window

Display Main Application:

Show a new window with a form to submit data, a button to view data, and an exit button.

Submit Data:

Display input fields for the user to input data.

When the "Submit" button is clicked:

Validate the input (ensure it is in the correct format).

If valid, save the data to a local database or file.

View Data:

Display a list or table of stored data when the "View Data" button is clicked.

Exit Application:

When the "Exit" button is clicked, confirm the exit request with a prompt.

If confirmed, close the application.

Explanation of the Algorithm:

1. initialize_application(): Initializes the GUI environment and sets up the main window.
2. show_login_screen(window): Displays a login screen with fields for entering the username and password. It validates the user input and proceeds to the main window if successful.
3. show_main_window(): Displays the main application window with buttons to submit data, view saved data, and exit the application.

- submit_data(): Allows the user to enter data and submit it. The data is validated and saved to a file (or database).
- view_data(): Displays previously stored data in a message box.
- exit_application(): Confirms the user's intention to exit the application.

4. save_data(data): A helper function that saves the input data to a file (you can replace it with a database interaction as needed).

Key Concepts in the Algorithm:

- Event-Driven Programming: The application uses Tkinter's event-driven model, where the application responds to user actions (like button clicks).
- Data Validation: The login credentials are validated, and the data entered by the user is checked before it is saved.
- Data Persistence: The example saves data to a local file (data.txt). You can extend this to use a database or other storage mechanisms.
- User Interaction: The application provides feedback to the user through pop-up messages (e.g., success or error messages) using messagebox.

Conclusion:

This algorithm outlines the basic structure of a modern GUI Windows software application built with Python. It handles user login, data submission, viewing stored data, and exiting the application. The example uses **Tkinter** for the GUI, and you can extend it to incorporate additional features such as advanced data processing, interaction with external APIs or databases, and more complex user interfaces.

CHAPTER: - 7
TEST CASES AND TEST REPORT

Test Case ID	OA_001	**Test Case Description**	Test the Login functionality for students		
Created By	Vishwajeet Wagh	**Reviewed By**	Vivek Shende	**Version**	1.0

QA Tester's Log	Initial Version

Tester's Name	Vivek Shende	**Date Tested**	20-Oct-2024	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Verify Window Title
2	Verify Button Functionality
3	Verify RadioButton Behavior

S #	Test Data
1	User id = Vivek001
2	Pass = Vivek1234

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Launch the application.	The window title should be "My Tkinter Application".	As Expected	Pass
2	Check the window title.	The window title should be "My Tkinter Application".	As Expected	Pass
3	Locate and click the "Submit" button.	After clicking the "Submit" button, the label should display "Submission Successful".	As Expected	Pass

Test Case ID	OA_002	**Test Case Description**	Test the functionality of taking an aptitude test.		
Created By	Yash Pawar	**Reviewed By**	Vivek Shende	**Version**	1.0

QA Tester's Log Initial Version

Tester's Name	Vivek Shende	**Date Tested**	29-Oct-2024	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Verify Window Resizing
2	Verify Menu Item Functionality

S #	Test Data
1	TestID = Menu_Function_01
2	
3	
4	

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Launch the application.	A file dialog should open to allow file selection	Verify that the menu item triggers the correct functionality.	Pass
2	Click on the "Open" menu item	A file dialog should open to allow file selection	As Expected	Pass
3	Check if a file dialog appears or the intended action occurs	A file dialog should open to allow file selection	As Expected	Pass

Test Case ID	OA_003	**Test Case Description**	Test the functionality of adding a new question to a subject by an admin.		
Created By	Yash Hasbe	**Reviewed By**	Vivek Shende	**Version**	1.0

QA Tester's Log Initial Version

Tester's Name	Vivek Shende	**Date Tested**	8-Nov-2024	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Verify Message Box Display
2	Verify Window Resizing
3	
4	

S #	Test Data
1	TC_08_Window_Resize
2	TC_07_MessageBox
3	Verify that the message box displays the correct message
4	Verify that resizing works as expected.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Launch the application.	Open application	As Expected	Pass
2	Click the "Show Message" button.	The message box should appear with the message	As Expected	Pass
3	Resize the window by dragging the corner.	The window should resize properly	Verify that resizing works as expected.	Pass
4	Verify that the layout adjusts correctly (i.e., no widget overlaps or cuts off).	The window should resize properly, and the layout should adjust without any widgets overlapping	Verify that resizing works as expected.	Pass

Test Case ID	OA_004	**Test Case Description**	Test the teacher login process, admin approval of the request, and assigning salary details.		
Created By	Vaibhav Ukande	**Reviewed By**	Vivek Shende	**Version**	1.0

QA Tester's Log Initial Version

Tester's Name	Vivek Shende	**Date Tested**	21-Nov-2024	**Test Case (Pass/Fail/Not Executed)**	Pass

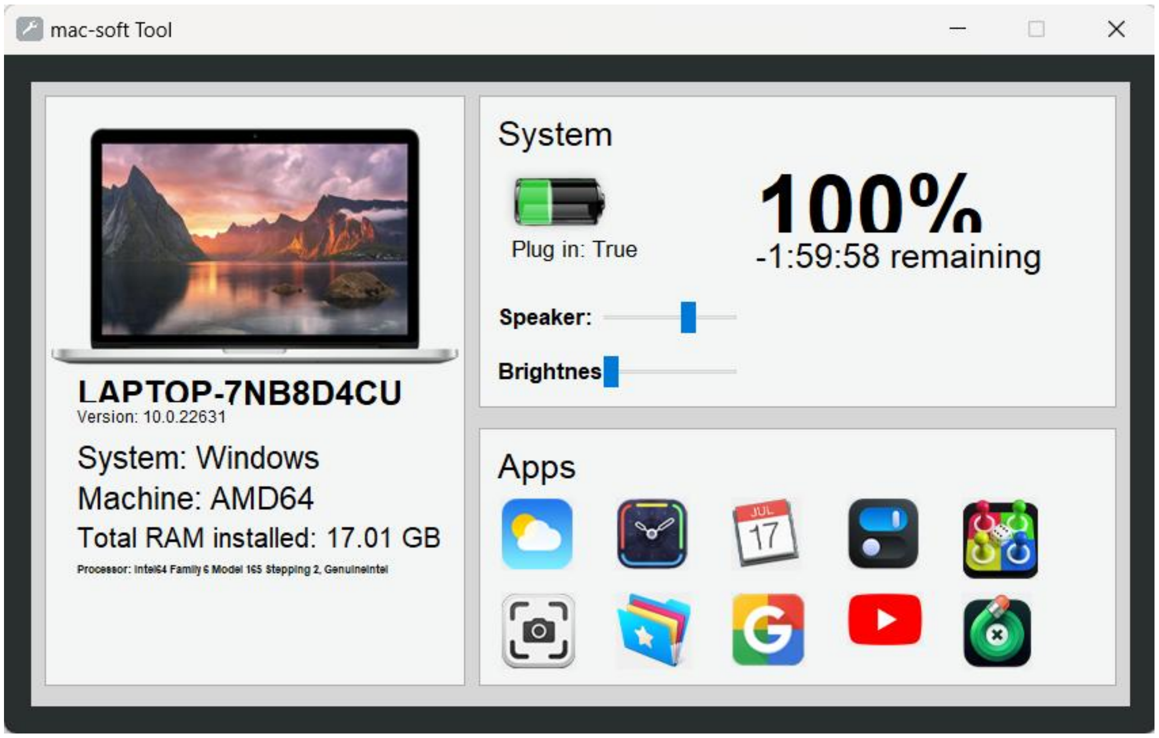
S #	**Prerequisites:**
1	Verify Dark/Light Theme Switching
2	Verify Invalid Input Handling in Textbox

S #	**Test Data**
1	TC_09_Theme_Switch
2	TC_10_Invalid_Input
3	Verify that the theme switch works and the widgets adjust accordingly

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Click the "Switch Theme" button Check if the theme switches from light to dark or vice versa	The application theme should toggle between dark and light mode	Verify that the theme switch works and the widgets adjust accordingly.	Pass
2	Click "Submit" and verify if an error message or validation occurs	The application should show an error message like "Invalid input, please try again	Verify that the error message is displayed correctly.	Pass

CHAPTER: 8
PROJECT SCREENSHOTS

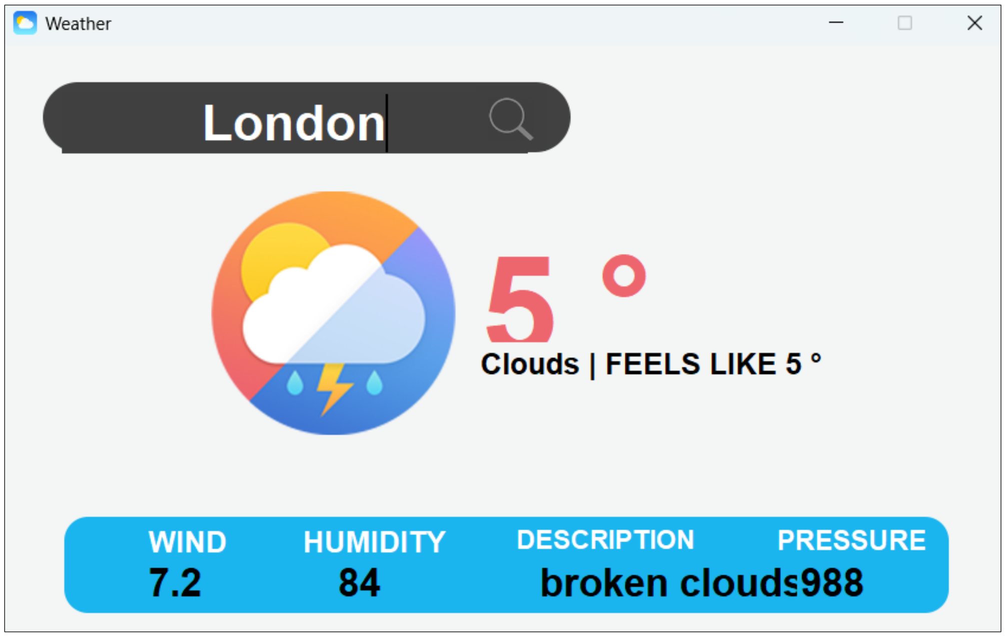
Interface :



LAPTOP-7NB8D4CU
Version: 10.0.22631
System: Windows
Machine: AMD64
Total RAM installed: 17.01 GB
Processor: Intel64 Family 6 Model 165 Stepping 2, GenuineIntel

Plug in: True
100%
-1:59:58 remaining

Icons Working :



London
5 °
Clouds | FEELS LIKE 5 °

WIND 7.2
HUMIDITY 84
DESCRIPTION broken clouds
PRESSURE 988

CHAPTER 9 :

CONCLUSION, FUTURE WORK, REFRENCE

Conclusion: Python provides a robust and versatile environment for developing modern GUI-based applications on Windows. Leveraging powerful libraries such as Tkinter, PyQt, wxPython, and Kivy, developers can create highly interactive and user-friendly interfaces that cater to a wide range of use cases, from simple tools to complex enterprise solutions.

Our project aims to redefine user interaction by leveraging contemporary design principles and advanced technologies. The application is meticulously crafted to enhance user experience through a sleek and responsive interface, streamlined workflows, and robust features tailored to meet both personal and professional needs.

Future Work:

1. Integration with AI/ML
2. Augmented Reality (AR) & Virtual Reality (VR)
3. Rich User Experience (UX)
4. Cloud Integration and Remote Access
5. Native Mobile App Development
6. Security and Privacy

References

1. **Author:** Steve Krug
   Name of book: "Don't Make Me Think"
   Page number : 399-458,

   Year of publication: 2003.

2. **Author:** Don Norman
   Name of book: *"The Design of Everyday Things"*

   Year of publication: 2008.