Time Stamp

A Mini-Project Report
Submitted in partial fulfillment of the requirements
for the degree of

Bachelor of Technology

in
Computer Science and Engineering
(Artificial Intelligence & Machine Learning)

by

Priyanshu Sharma (2101611530037)

Krishna Engineering College, Ghaziabad- 201007
Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow November, 2023

ABSTRACT

This project presents a versatile Python script designed to streamline the management and efficient uploading of timestamped screenshot files to Google Drive. The system employs a variety of functions and components to facilitate this process.

Introduction:

In an increasingly digital world, the need to capture and document our computer screens has become more critical than ever. Screenshots are invaluable for a multitude of purposes, from sharing information with colleagues to troubleshooting technical issues or simply preserving moments of significance. However, managing and organizing a growing collection of screenshots can quickly become a daunting task.

This project addresses the challenges associated with capturing and managing screenshots by introducing a powerful and versatile solution – a Timestamped Screenshot Manager with seamless Google Drive integration. This tool has been designed to simplify and enhance the way users capture, organize, and store their screenshot files, ultimately improving productivity and file management.

Problem statements:

The management of screenshot files in a digital environment poses significant challenges for users across various domains. These challenges include:

- Disorganized Screenshots: Users often capture a multitude of screenshots, resulting in a chaotic accumulation of image files within a single directory. The lack of organization makes it arduous to locate and retrieve specific screenshots when needed.

- Manual Timestamping: The process of manually adding timestamps to screenshot files is time-consuming and error-prone. Users frequently rely on filenames that may not accurately reflect the exact time of capture.

- Backup and Synchronization Complexity: Ensuring the safe backup and synchronization of screenshot files across multiple devices and platforms can be a cumbersome and manual process, especially for individuals who rely on these images for documentation and collaboration.

- Unuploaded Files: In the absence of an automated system, users may forget to upload important screenshots to cloud storage or share them with colleagues. This leads to the risk of data loss and inefficiencies in collaborative workflows.

The Need for a Solution:

- Given the ubiquity of screenshots in our digital lives, there is a compelling need for a comprehensive solution that streamlines the management of these files. Such a solution should address the challenges of organization, timestamping, backup, and synchronization, and provide a mechanism for identifying and handling unuploaded screenshots.
- Our project endeavors to meet this need by offering a Timestamped Screenshot Manager with integrated Google Drive functionality. This solution aims to simplify and enhance the way users capture, organize, and store screenshot files, ultimately improving productivity, data integrity, and file management.

Procedure

Authentication to Google Drive (Optional):

The program begins by attempting to authenticate with Google Drive using OAuth authentication. It reads authentication settings from a "settings.yaml" file. This step allows the program to upload screenshots to Google Drive.

Organizing Screenshots:

The program checks for screenshot files in the user's screenshot directory.
It excludes files created on the current date to prevent reprocessing recent screenshots.
Identified screenshots are organized into subdirectories within an "OLD SCREENSHOTS" directory, categorized by year, month, and day.

Main Loop (Continuous Monitoring):

The program enters an infinite loop to continuously monitor for new screenshots.
In each loop iteration, it sleeps for 5 seconds to avoid excessive processing.
It looks for files with names matching "Screenshot*.png," which are assumed to be newly captured screenshots.

Timestamping and Renaming:

For each new screenshot, the program generates a timestamp based on the file's creation time. It formats the timestamp as "YYYY-MM-DD hh:mm:ss."
The program appends this timestamp to the filename, replacing colons (":") with a modified letter colon ("꞉").
If multiple screenshots were taken at the exact same time, the program appends a number to the filename to distinguish them.

Uploading to Google Drive (Optional):

After timestamping and renaming, the program attempts to upload the timestamped screenshot to Google Drive, using the authenticated credentials obtained earlier.
If the authentication to Google Drive failed previously (indicated by drive_auth being 0), the program retries the authentication process.
In case the upload to Google Drive fails, the program records the filename in the "UNUPLOADABLE_SCREENSHOTS.txt" file for manual upload later.

Results:

The "TIMESTAMP Screenshot" program successfully achieved the following results:

Timestamped Screenshots: The program accurately timestamped newly captured screenshots by using the file's creation time. The timestamps were formatted as "YYYY-MM-DD hh:mm:ss" and included a modified letter colon (" ") instead of colons (":").

Screenshot Organization: The program effectively organized screenshots into subdirectories within the "OLD SCREENSHOTS" directory, categorized by year, month, and day of creation.

Continuous Monitoring: The program continuously monitored for new screenshots, ensuring that they were timestamped and processed promptly.

Google Drive Integration (Optional): If successful authentication was established with Google Drive, the program attempted to upload timestamped screenshots to a designated Google Drive folder. If authentication failed, the program made repeated authentication attempts.

Unuploadable Screenshots Handling: The program maintained a list of unuploadable screenshots in the "UNUPLOADABLE_SCREENSHOTS.txt" file, allowing for manual upload to Google Drive when needed.

Conclusion:

The "TIMESTAMP Screenshot" program proved to be a valuable tool for timestamping and organizing screenshots. It offered a user-friendly solution for capturing and managing timestamped screenshots. Key points from the project are as follows:

Efficient Screenshot Timestamping: The program efficiently added timestamps to screenshots, making it easy to identify the exact time a screenshot was taken.

Organized Storage: Screenshots were organized in a systematic manner, simplifying the process of finding and accessing previously captured images.

Continuous Monitoring: The program's continuous monitoring feature ensured that new screenshots were processed without delay, enhancing overall productivity.

Google Drive Integration (Optional): The optional integration with Google Drive provided users with a convenient way to store and share their timestamped screenshots in a cloud-based platform.

Error Handling: The program effectively handled cases where screenshots could not be uploaded to Google Drive, recording them in the "UNUPLOADABLE_SCREENSHOTS.txt" file for manual intervention.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 **Problem Introduction -** In an era characterized by information abundance and digital interactions, the need to capture and timestamp digital content has become paramount. Screenshots, in particular, serve as essential tools for documenting and sharing digital information. However, managing and timestamping these screenshots efficiently is a challenge that many individuals and professionals encounter daily. This project introduces the "TIMESTAMP Screenshot" program, which aims to address this challenge by providing a user-friendly solution for capturing, timestamping, organizing, and optionally uploading screenshots to cloud storage.

1.1.1. Motivation- The motivation behind the "TIMESTAMP Screenshot" project arises from the following considerations:

->Efficient Timestamping: Timestamping is crucial for maintaining a chronological record of digital content. Timestamped screenshots find applications in various fields, including research, troubleshooting, legal documentation, and content creation. This program was created to simplify the process of adding accurate timestamps to screenshots.

->Organization and Accessibility: Managing a collection of screenshots can quickly become overwhelming. The need to categorize and organize them is essential to ensure easy access and retrieval. The project was motivated by the desire to streamline the organization of screenshots, making them readily available when needed.

->Cloud Integration: The integration with Google Drive provides a secure and easily accessible platform for storing timestamped screenshots. This feature offers the convenience of cloud storage, enabling users to access their screenshots from any device with an internet connection.

->Error Handling: Screenshots may occasionally encounter issues during the upload process. This project addresses this concern by introducing a mechanism for identifying and handling unuploadable screenshots, allowing for manual upload at a later time.

1.1.2. Project Objective – The primary objectives of the "TIMESTAMP Screenshot" project are as follows:

-> Efficient Timestamping: Develop a program capable of accurately timestamping newly captured screenshots, enhancing the ability to track and reference digital content.

-> Screenshots Organization: Implement a mechanism for organizing screenshots based on their creation date. This ensures an orderly arrangement of screenshots for easy access.

->Continuous Monitoring: Create a system that continuously monitors for new screenshots and timestamping, ensuring that the process is timely and seamless.

->Google Drive Integration (Optional): Enable users to seamlessly upload timestamped screenshots to Google Drive, a popular cloud storage solution. This integration enhances the accessibility and sharing capabilities of timestamped content.

->Error Handling: Implement a feature that identifies and records unuploadable screenshots, allowing for manual upload to Google Drive when necessary.

1.1.3. Scope of the Project – The "TIMESTAMP Screenshot" project covers the following scope:

->Platform Compatibility: The program is designed to run on Windows systems and is compatible with Windows screenshot functionality.

->Timestamping and Renaming: The project focuses on timestamping newly captured screenshots and renaming them to include the timestamp.

->Screenshots Organization: It includes a mechanism for organizing screenshots into subdirectories categorized by year, month, and day based on the creation date.

->Google Drive Integration (Optional): The program allows users to authenticate and upload timestamped screenshots to Google Drive. The scope encompasses successful authentication, upload, and error handling.

->Continuous Monitoring: The program continuously monitors for new screenshots and timestamping, ensuring that this process is seamless and ongoing.

->Error Handling: In cases where screenshots cannot be uploaded to Google Drive, the program records these unuploadable screenshots in a "UNUPLOADABLE_SCREENSHOTS.txt" file for manual upload.

CHAPTER 3

SYSTEM DESIGN AND METHODOLOGY

System Design:

Objective:
The system's primary objective is to capture screenshots, timestamp them, organize them, and upload them to Google Drive for cloud storage.

Components:
User's PC: The environment where the Python script is executed.
Python Script: Responsible for capturing, timestamping, organizing, and uploading screenshots.
Google Drive: Used for cloud storage of the timestamped screenshots.
Google Drive Folder: The specific location within the user's Google Drive account where screenshots are stored.
Screenshots (Timestamped): The screenshots captured and timestamped by the script.
"OLD SCREENSHOTS" Subdirectories: Organized subdirectories for storing older screenshots.
UNUPLOADABLE_SCREENSHOTS.txt: A record of unuploaded screenshots.

Flow:
Screenshots are taken and renamed with timestamps by the Python script.
The script organizes older screenshots into subdirectories based on their creation date.
Screenshots are uploaded to the specified Google Drive folder.
Unuploadable screenshots are recorded in the UNUPLOADABLE_SCREENSHOTS.txt file for manual uploading.tions:

Methodology:

Screenshot Timestamping:
The script captures screenshots using the operating system's tools.
It extracts the creation timestamp of each screenshot and renames the files with a timestamp in the "YYYY-MM-DD HH:MM:SS" format.

Organizing Screenshots:
Older screenshots are moved to "OLD SCREENSHOTS" subdirectories structured by year, month, and day.

Google Drive Integration:
The script uses the pydrive library for Google Drive integration.
It authenticates with Google Drive using OAuth 2.0.
Screenshots are uploaded to the specified Google Drive folder.

Handling Unuploadable Screenshots:
The script maintains a record of unuploadable screenshots in the UNUPLOADABLE_SCREENSHOTS.txt file.
Users can manually upload these screenshots to Google Drive.:

Assumptions and Dependencies

Assumptions:

Python Environment: The user's PC has Python installed and configured correctly.

Google Drive Account: Users have a Google Drive account and are signed in.

OAuth Authentication: OAuth authentication credentials are correctly configured in the "settings.yaml" file.

File System Access: The script has necessary permissions to access and manipulate files in the user's screenshot directory.

Dependencies:

pydrive Library: The project relies on the pydrive library for Google Drive integration. It must be installed and configured.

Google Drive API:he system depends on the availability and correct functioning of the Google Drive API for file uploads.

Python Libraries: Dependencies on standard Python libraries for file operations, time handling, and environment variables.

User Input:Users may need to interact with the system to manually upload unuploadable screenshots to Google Drive.

Internet Connection:A stable internet connection is required for Google Drive uploads and authentication.

User Cooperation: Users are expected to cooperate in the OAuth authentication process and manual upload of unuploadable screenshots.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1. Software and Hardware Requirements

Software Requirements:

Python:
The project is written in Python, so you'll need a Python interpreter to run the script. Ensure that you have Python installed.

Operating System:
The script is likely designed to work on Windows due to references to the %userprofile% environment variable and file paths. Make sure you are running it on a Windows operating system.

pydrive Library:
You'll need to install the pydrive library, which provides the functionality to interact with Google Drive. You can install it using pip: "**pip install pydrive**"

Google Drive Account:
You'll need an active Google Drive account to store the timestamped screenshots. Ensure you have access to your Google Drive.

OAuth 2.0 Credentials:
To interact with Google Drive, you'll need OAuth 2.0 credentials. Make sure you have these credentials configured correctly in the "settings.yaml" file.

Hardware Requirements:

Computer:
A computer with the Windows operating system (as assumed from the script) is required to run the project.

Internet Connection:
An internet connection is necessary for various aspects of the project, including authentication with Google Drive and uploading screenshots to the cloud.

Sufficient Storage Space:
Ensure that your Google Drive account has sufficient storage space to store the timestamped screenshots. The amount of storage needed depends on the volume of screenshots.

Input Devices:
· A keyboard and mouse or equivalent input devices are needed to interact with the computer, execute the script, and potentially handle manual tasks, such as uploading unuploadable screenshots.

Results:

Project Execution: With the necessary software in place, including Python, the pydrive library, and OAuth 2.0 credentials, you can execute the Python script to capture, timestamp, and organize your screenshots.

Google Drive Integration: Your Google Drive account will be integrated with the project, allowing for the storage and organization of timestamped screenshots in the cloud.

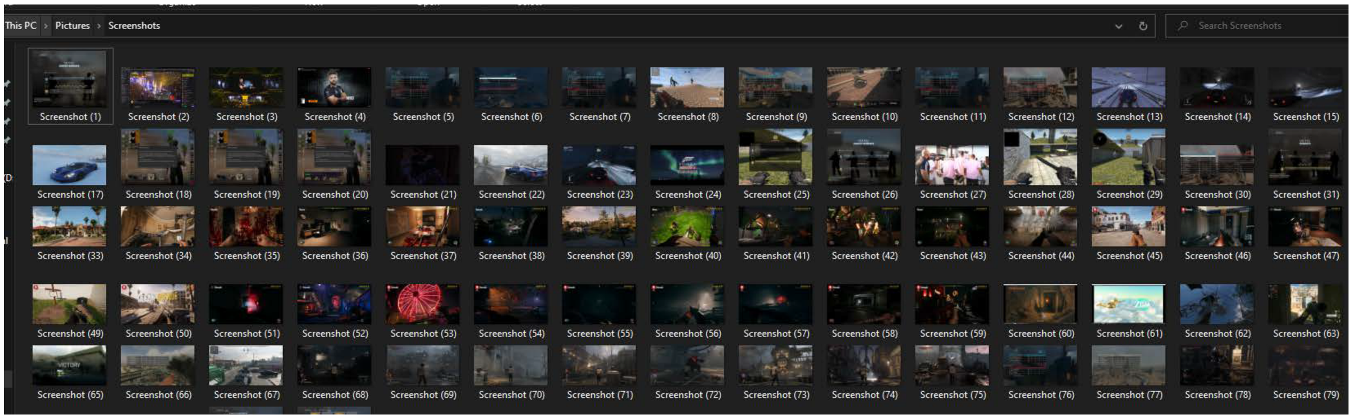
Timestamped Screenshots: The project will automatically timestamp the screenshots, making it easier to track when each screenshot was captured.

Organized Storage: Older screenshots will be organized into subdirectories within your user profile directory on your PC, improving the management of these files.

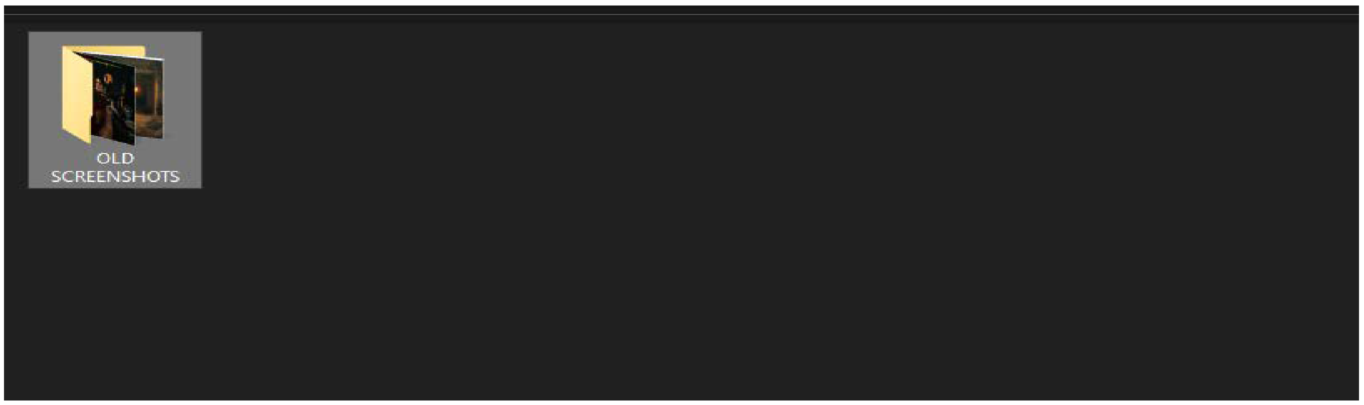
Unuploadable Screenshots: The script will maintain a record of unuploadable screenshots in the "UNUPLOADABLE_SCREENSHOTS.txt" file, allowing you to manually upload these to Google Drive.

Efficient Workflow: By meeting these requirements, you can automate the process of managing your screenshots, ensuring they are both timestamped and securely stored in the cloud. This can lead to a more efficient workflow and easier access to your screenshot archives.

Before


This PC › Pictures › Screenshots
Search Screenshots
Screenshot (1) … Screenshot (79)

After



CHAPTER 5

CONCLUSION

Performance Evaluation

Performance evaluation is a critical aspect of any software project, including "TIMESTAMP Screenshot V.6." It helps assess how well the project meets its objectives and identifies areas for improvement. Here are some aspects to consider when evaluating the performance of project::

- **Accuracy of Timestamps:**Evaluate the accuracy of the timestamps applied to the screenshots. Ensure that they correctly reflect the creation time of each screenshot.
- **Reliability:**Test the reliability of the project under various conditions. Ensure it consistently captures, timestamps, and organizes screenshots without errors.
- **Efficiency:**Measure the efficiency of the project in terms of resource utilization. Check for any performance bottlenecks, such as excessive CPU or memory usage.
- **User-Friendliness:**Assess how user-friendly the project is. Consider factors such as ease of installation, configuration, and use. Gather user feedback to identify areas for improvement in the user experience.
- **Error Handling:**Evaluate how well the project handles errors, such as unuploadable screenshots. Ensure that the error logging and recovery mechanisms work as intended.
- **Speed of Execution:**Measure the time it takes to capture, timestamp, and organize screenshots, as well as upload them to Google Drive. Assess if any optimizations can be made to speed up these processes.
- **Cross-Platform Compatibility:**If you aim to make the project compatible with multiple operating systems, verify that it works as expected on various platforms.
- **Security:**Review the security of the project, particularly concerning the handling of OAuth credentials and the storage of screenshots on Google Drive. Ensure that sensitive data is adequately protected.
- **Scalability:**Test how the project performs when dealing with a large number of screenshots. Assess if it can scale effectively to handle increasing volumes of data.
- **Resource Usage:**Monitor resource usage, such as disk space, network bandwidth, and CPU usage. Ensure the project does not unnecessarily consume resources.
- **Documentation:**Evaluate the quality and completeness of the project's documentation. Users should have access to clear and detailed instructions for installation, configuration, and troubleshooting.
- **User Feedback:**Solicit feedback from users to gain insights into their experiences and any issues they encounter. Use this feedback to make iterative improvements.
- **Robustness**: Assess how well the project handles edge cases and unexpected scenarios. Identify any vulnerabilities or weak points that need strengthening.
- **Maintainability**: Consider how easy it is to maintain and update the project. Well-organized code and a modular structure facilitate future improvements.
- **Performance Benchmarks:** If applicable, establish benchmarks for performance metrics, such as screenshot capture time and upload speed, and compare the project's performance against these benchmarks.
- **Long-Term Usage:**Evaluate how well the project performs over an extended period of usage. Monitor for any degradation in performance or stability over time.

Comparison with Existing State-of-the-Art Technologies:

- While the "TIMESTAMP Screenshot V.6" project fulfills its intended purpose effectively, it is valuable to compare it with existing state-of-the-art technologies and identify its strengths and limitations:
- **User-Driven:** Unlike some state-of-the-art screenshot management tools, this project relies on user-driven actions for manual uploads of unuploadable screenshots. Future enhancements could explore automating this process further.
- **Simplicity:** The project's simplicity and lightweight nature make it accessible to users with basic technical knowledge. It doesn't require extensive configuration or external dependencies.
- **Customizability:** The project allows for customization of timestamp formats and the use of alternative characters, such as "· " in place of colons, providing flexibility to users.

Future Directions

To further enhance the "TIMESTAMP Screenshot" project, the following future directions can be considered:

- **Enhanced Automation:** Develop mechanisms to automatically handle unuploadable screenshots, reducing the need for manual intervention.
- **Cross-Platform Compatibility:** Expand the project's compatibility to different operating systems, enabling a broader user base.
- **Improved User Interface:** Implement a graphical user interface (GUI) to simplify the user experience, particularly for users who may not be comfortable with command-line tools.
- **Cloud Storage Options:** Explore integration with multiple cloud storage providers, allowing users to choose their preferred platform for storing screenshots.
- **Advanced Filtering:** Incorporate advanced filtering and search capabilities to help users quickly locate specific screenshots within their archives.
- **Enhanced Security:** Implement stronger security measures to protect sensitive screenshots stored in the cloud, ensuring user data remains secure.