PROJECT ABSTRACT

Desktop Screen Time and App Usage Tracker

A desktop screen time tracker with per-app usage analytics, focus streaks, weekly trends and category budgets — 100% local, no cloud. Delivered built-to-order with the full application, installers, report, PPT and viva kit.

01. Title

Desktop Screen Time and App Usage Tracker — record which applications hold the foreground and for how long, then turn the data into hourly charts, top-app rankings, focus streaks and week-over-week trends.

02. Introduction / Background

People systematically underestimate screen time and misremember where it went. Phones now ship usage dashboards, but desktops — where students and professionals do their real work — have no equivalent. A desktop tracker watches the active window, logs per-application foreground time, classifies apps into categories, and presents the day as charts: hourly stacked bars, productive vs distracting splits, longest focus streaks and weekly comparisons.

03. Problem Statement

Without measurement there is no feedback loop for digital habits. Existing desktop tools are either cloud-based (uploading behavior data) or crude timers. This project builds a privacy-first desktop tracker: all data stays in a local SQLite database, the service is event-driven with negligible overhead, and the analytics answer 'where did my day go' with charts instead of guilt.

04. Objectives

- Track the foreground window per OS (Win32, NSWorkspace, X11) with process attribution via psutil.
- Sessionize samples into per-app usage with automatic categorization and user overrides.
- Render hourly activity charts, top-app rankings, focus-streak detection and weekly trend comparisons.
- Support per-category daily budgets with tray nudges when distracting time runs high.
- Keep 100% of data local; provide CSV export; ship installers and full viva documentation.

05. Proposed Methodology

A lightweight tracker service samples the foreground window every 5 seconds (configurable 1–60 s) and attributes it to the owning process. Consecutive samples merge into sessions. A rules engine categorizes executables (development, browsing, entertainment, communication, etc.) with user overrides. An analytics layer aggregates sessions into hourly buckets, daily totals and streaks, served to a Qt dashboard with custom canvas charts. Goals are evaluated

live against category budgets.

06. System Architecture / Working

OS window hooks → sampler (5 s) → process attribution (psutil) → sessionizer → categorizer (rules + overrides) → SQLite store → analytics aggregator → Qt dashboard (hourly chart, top apps, weekly trends, goals). All components run locally; the app makes zero network calls.

07. Hardware / Software Requirements

- A Windows 10/11, macOS 13+ or Ubuntu 22.04+ machine.
- Python 3, PyQt6, SQLite, psutil, OS-specific hook libraries, PyInstaller.
- Expected overhead: under 1% CPU and roughly 40 MB RAM.

08. Expected Outcomes

A working tracker + dashboard with hourly, daily and weekly analytics, goals and nudges — plus installers, a categorization ruleset, a project report, PPT and viva Q&A.

09. Applications

- Students auditing study vs distraction time during exam preparation.
- Freelancers producing honest time breakdowns for clients.
- Teams running voluntary digital-wellbeing programs.

10. Future Scope

- Browser-extension for per-website attribution.
- Optional encrypted multi-device aggregation.
- Focus-mode that dims distracting apps when budgets are exceeded.