System Monitor Project Report

Project Name: Kali Smart System & Study Assistant Manager **File:** temp.py

Date: December 12, 2025 **Language:** Python 3 **Framework:** Tkinter GUI

Project Members:-
M. Ahmed (196)Ameer
Hamza Athar (040)
Haroon Mazhar (121)
Sameer Hassan (168)

📋 Executive Summary

This project is a comprehensive system monitoring and productivity assistant application designed for Kali Linux environments. It combines real-time system monitoring, security analysis, productivity tracking, and system management tools in a single dark-themed GUI interface.

🎯 Project Objectives

1. **Real-time System Monitoring**
   Monitor CPU, RAM, disk usage, and network traffic

2. **Security Analysis**
   Track failed login attempts and detect dangerous commands

3. **Productivity Tracking**
   Analyze shell history for study/work patterns

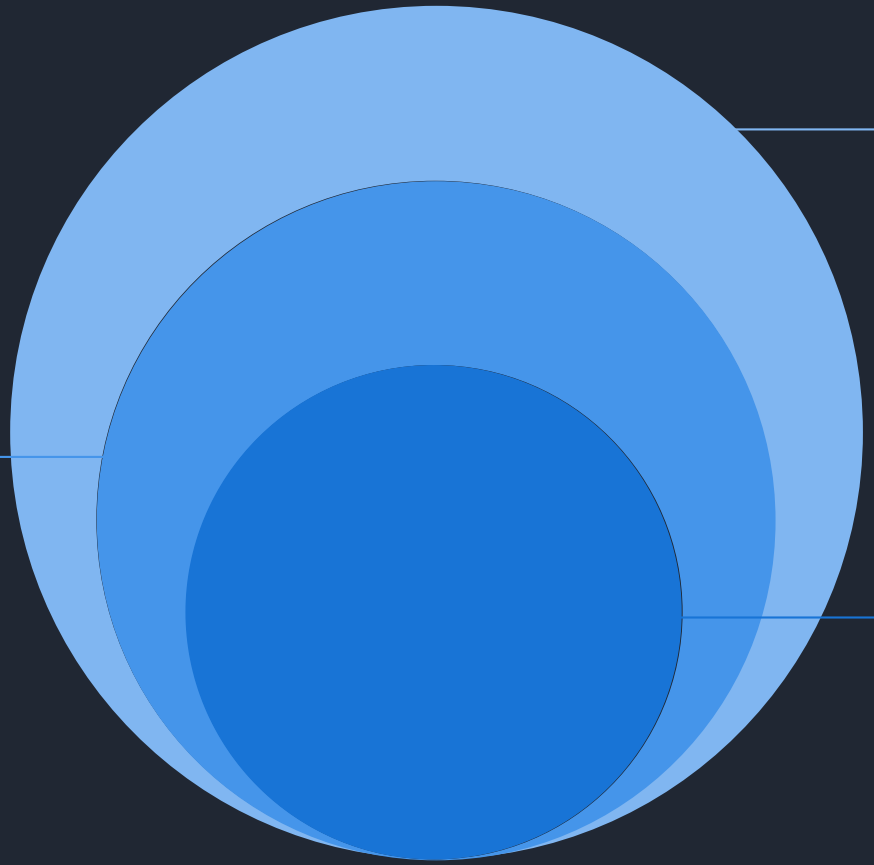
4. **Process Management**
   View and manage running processes

5. **Data Visualization**
   Provide live performance graphs

6. **System Health Assessment**
   Calculate overall system health scores

7. **Backup & Reporting**
   Generate reports and backup project files

🏗️ Architecture Overview



Data & GUI

Processing logic and Tkinter user interface

System Interface

Abstraction layer handling OS interactions

Core APIs

psutil and subprocess for system calls

🔧 Technical Specifications

Dependencies

- **Core Libraries**: tkinter, ttk, psutil
- **System Integration**: os, subprocess, socket
- **Data Structures**: collections.deque
- **File Operations**: filedialog, messagebox
- **Time Management**: time, datetime
- **Randomization**: random

Performance Characteristics

- **Refresh Rate**: 1.5 seconds auto-refresh
- **Memory Usage**: Circular buffers (60 samples max)
- **CPU Monitoring**: 0.2-second intervals
- **Network Monitoring**: Real-time byte counting

Platform Compatibility

- **Primary Target**: Kali Linux
- **Secondary Support**: Other Linux distributions
- **File System**: Unix-style paths (/, ~/.bash_history)

User Interface Design

Layout Structure

Color Scheme

- **Background**: Dark theme (#111111, #202020)
- **Success Indicators**: Green (#00cc66)
- **Warning Indicators**: Orange (#ff9900)
- **Critical Indicators**: Red (#ff3333)
- **UI Elements**: Gray tones (#303030, #444444)

Visual Components

- **System Info Labels**: Real-time text displays
- **Performance Graphs**: Line charts with grid backgrounds
- **Process Table**: Sortable tree view with color coding
- **Control Buttons**: Styled action buttons
- **Status Panels**: Information displays with color coding

📊 Feature Analysis

Core Features

CPU Monitoring	psutil.cpu_percent()	✅ Complete
RAM Monitoring	psutil.virtual_memory()	✅ Complete
Disk Usage	psutil.disk_usage()	✅ Complete
Network Traffic	psutil.net_io_counters()	✅ Complete
Process Management	psutil.process_iter()	✅ Complete
Security Analysis	Log file parsing	✅ Complete
Graph Visualization	Canvas drawing	✅ Complete
Report Generation	File export	✅ Complete

Advanced Features

Health Scoring	System performance rating	Mathematical algorithm
Safety Checking	Dangerous command detection	Pattern matching
Study Tracking	Shell history analysis	File parsing + statistics
Auto-refresh	Live data updates	Timer-based callbacks
Data Export	System reports	File dialog + formatting
Project Backup	Compressed archives	tar.gz creation

🛡️ Security Features

Security Monitoring

- **Failed Login Detection**
  - Monitors /var/log/auth.log
  - Fallback to journalctl commands
  - Counts authentication failures
- **Dangerous Command Detection**
  - Scans shell history for risky commands
  - Pattern matching for destructive operations
  - Monitors recent command history (80 commands)
- **System Health Assessment**
  - Combines performance and security metrics
  - Penalty system for high resource usage
  - Security incident impact on health score

Dangerous Command Patterns

```
dangerous_keywords = [
 "rm -rf /", # System deletion
 "mkfs", # Filesystem formatting
 ":(){ :|:& };:", # Fork bomb
 "chmod 777", # Insecure permissions
 "dd if=", # Disk operations
]
```

📈 Performance Monitoring

Real-time Metrics

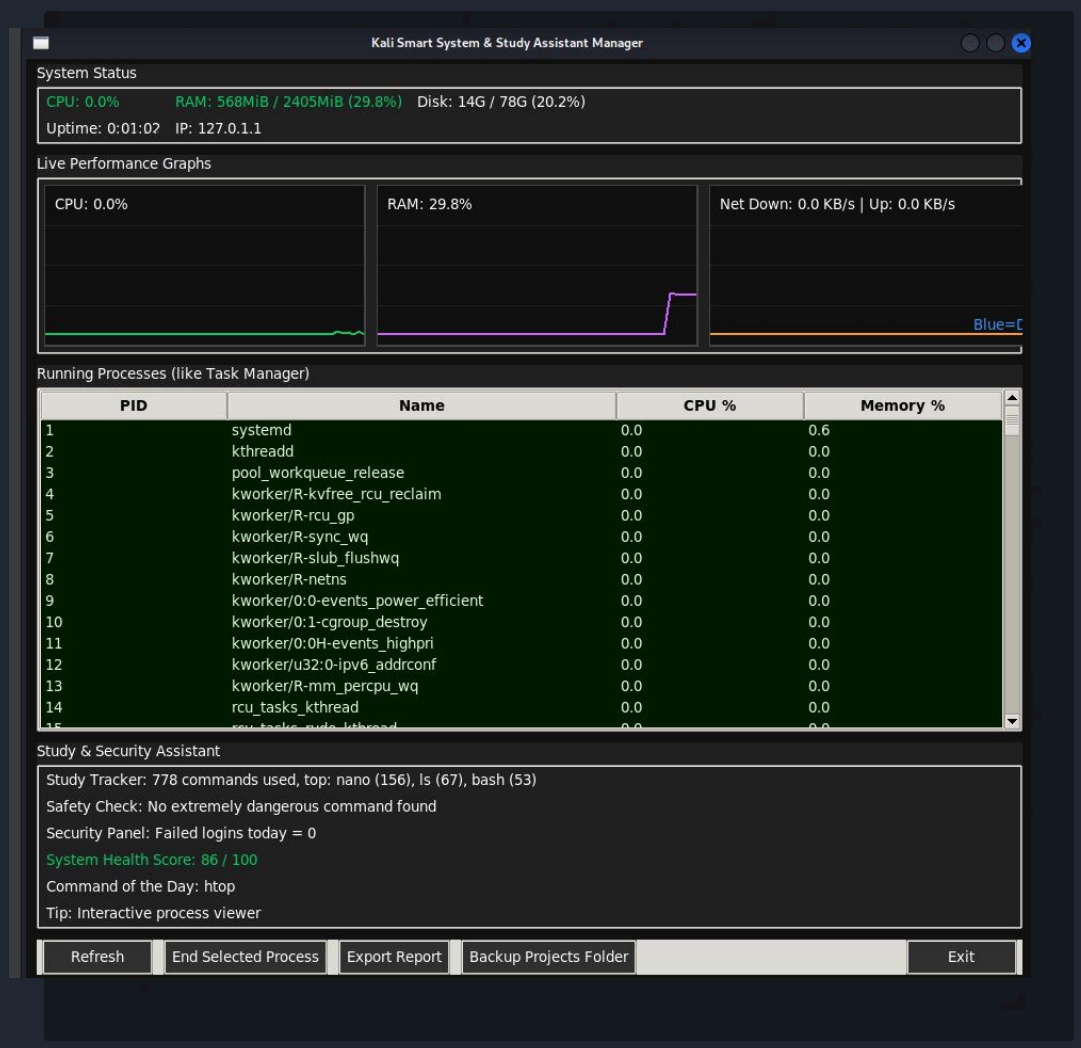
- **CPU Usage**: Percentage utilization with 0.2s sampling
- **Memory Usage**: Physical RAM consumption tracking
- **Disk Space**: Available storage monitoring
- **Network Traffic**: Upload/download rates in KB/s
- **Process Activity**: Individual process resource consumption

Historical Data

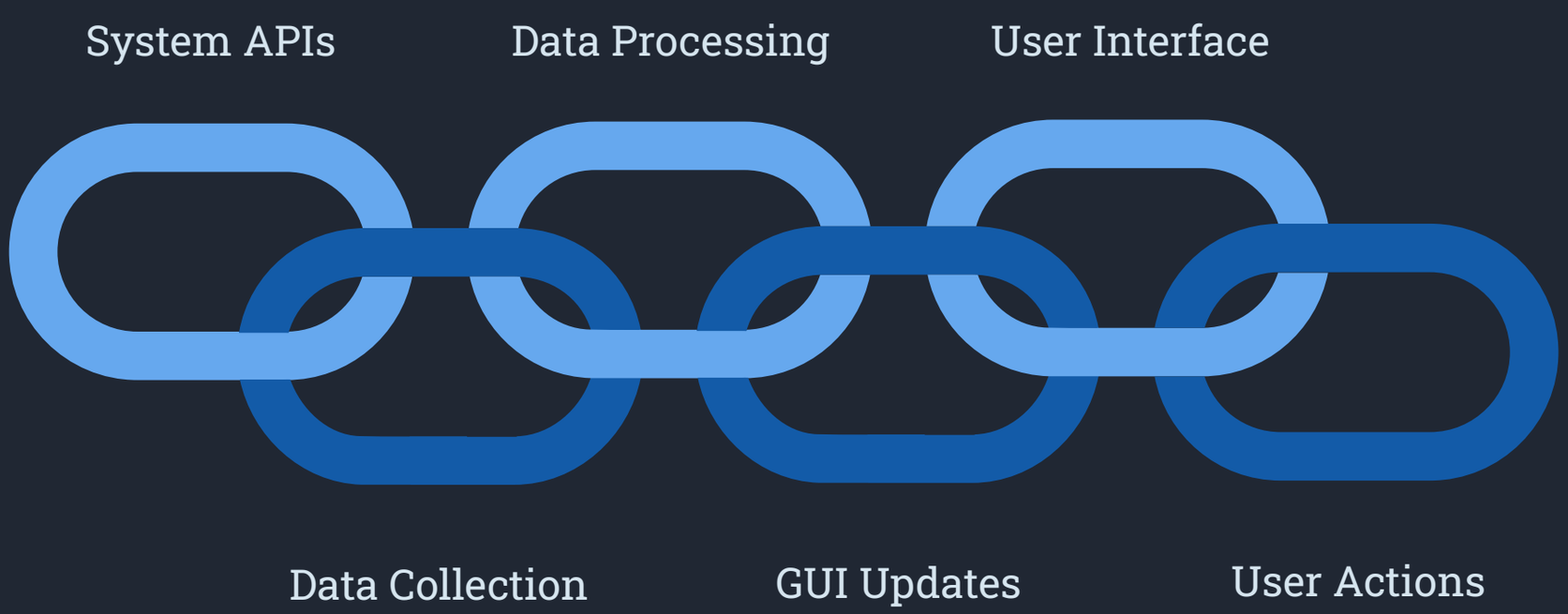
- **Buffer Size**: 60 samples (approximately 90 seconds)
- **Data Structure**: Circular buffers using collections.deque
- **Visualization**: Real-time line graphs with grid overlays

Color-coded Alerts

```
def _color_for_usage(value):
    if value < 50:
        return "#00cc66" # Green: Safe
    elif value < 80:
        return "#ff9900" # Orange: Warning
    else:
        return "#ff3333" # Red: Critical
```



🔄 Data Flow Architecture



Update Cycle

1. **Timer Trigger** (1.5s intervals)
2. **Data Collection** (system metrics)
3. **Processing** (calculations, formatting)
4. **GUI Update** (labels, graphs, tables)
5. **User Interaction** (button clicks, selections)
6. **Action Execution** (system commands)