Raspberry Pi System Monitor

**Project Description:** Create a bash script for a Raspberry Pi that functions as a system monitor. The script will gather information about the Raspberry Pi's operating system, file system, software, and basic Linux commands. It will present the information in a user-friendly format and include the ability to launch web browsers or applications.

**Skills/Objectives:**

1. **Operating Systems:** 
   * Understand the basics of the Raspberry Pi operating system (e.g., Raspbian).
   * Interact with the operating system using commands.
   * Understand the difference between Operating Systems
2. **File Systems:** 
   * Explore and display information about the file system on the Raspberry Pi.
   * Provide details about storage capacity, file types, and directory structures.
3. **Software:** 
   * Identify installed software on the Raspberry Pi.
   * Extract information about software versions and dependencies.
4. **Applications:** 
   * Create a menu-driven system to launch basic applications.
   * Allow users to interact with installed applications through the script.
5. **Web Browsers**:
   * Include functionality to launch a web browser from the script.
   * Display basic information about the default web browser.
6. **Software Licensing**:
   * Include information about the licensing of installed software.
   * Provide details on open-source licenses for relevant software.
7. **Basic Linux Commands**:
   * Utilize and explain fundamental Linux commands within the script.
   * Provide a help section that explains how to use the script.

**Project Phases:**

1. **Planning:**
   * Define the scope and objectives of the project.
   * Identify the key features and functionalities of the system monitor script.
   * Determine the Linux commands and utilities to be used for gathering system information.
   * Specify the structure of the menu-driven interface.
   * Consider potential challenges and how to address them (e.g., handling errors, ensuring user authentication).
2. **Design:**
   * Create a high-level overview of the script's architecture and flow.
   * Define the structure of the script, including functions and main execution flow.
   * Design the menu interface with clear options for users.
   * Plan how the script will gather and display information about the operating system, file system, software, and applications.
   * Specify how launching web browsers and applications will be implemented.
   * Design any additional features, such as real-time monitoring or user authentication.
3. **Testing and Feedback:**
   * Develop the initial version of the script based on the design.
   * Conduct initial testing to ensure basic functionality.
   * Collect feedback from users or test cases to identify potential improvements or issues.
   * Iterate on the script based on feedback, addressing any identified issues or incorporating suggested improvements.
   * Perform comprehensive testing, including edge cases and error scenarios.
4. **Documentation:**
   * Create comprehensive documentation for the script.
   * Include a README file with information on how to run the script, its dependencies, and any setup requirements.
   * Provide detailed documentation for each function and section of the script.
   * Document the purpose and usage of Linux commands used in the script.
   * Include a user guide that explains the menu options, how to interpret the displayed information, and any additional features.
   * Document any customization options and guidelines for users who want to modify the script.
   * Consider adding a troubleshooting section to address common issues.

**Python Libraries:**

You can use Python libraries like tkinter, subprocess, os, psutil and requests for creating a graphical user interface (GUI) and generating displays, viewing internal processes, etc..