**High-Level Design Document: Laptop Camera Component Testing**

---

# 1. Introduction

The goal of this document is to define a structured and scalable approach to testing a laptop's **components,** starting with the **camera**. The system will be designed to validate component's functionality, identify potential failures, and ensure a robust testing framework that can be extended to other hardware components in the future.

This testing system will be implemented using **Python** on **Ubuntu (20.04)**, leveraging `v4l2-ctl` for camera control and `fswebcam/GStreamer` for image and video capture.

---

# 2. Test Categories

The testing framework will include:

## 2.1 Functional Testing

✅ **Camera Availability**

- Verify that the camera **is available by default.**

✅ **Feature Settings**

- Validate **brightness, contrast, backlight compensation, and sharpness adjustments**.

✅ **Capture Image**

- Capture an image and verify it was **saved successfully**.

✅ **Video Recording**

- Record a video and validate **file integrity**.

✅ **Multiple Camera Handling**

- Identify and select **external webcams** when connected.

---

### 2.2 Negative Testing

❌ **Invalid Feature Inputs**

- Attempt to access the camera setting **invalid brightness, contrast and other features** .

❌ **Capture When Camera Unavailable**

- Attempt to **capture while another application is using the camera**.

❌ **Recording When Camera Unavailable**

- Attempt to video **record while another application is using the camera**.

---

### 2.3 Edge Cases

⚡ **Fast Camera Switching**

- Toggle between **multiple cameras rapidly** and check response time.

⚡ **Overloaded System**

- Run tests **while the CPU is under heavy load** to check stability.

---

## 3. Physical Setup Considerations

The "physical setup" refers to:

- **Internal vs. External Cameras**: If multiple cameras exist, ensuring correct selection.

---

## 4. Logs

The testing framework must include an **logging system** to:

- Print **test execution results**.
- Capture **error messages** and failures.

---

# 5. Implementation Milestones

🔹 **Phase 1:**

- Set up the environment (Ubuntu + Python).
- Basic camera detection using `v4l2-ctl`.

🔹 **Phase 2:**

- Implement **image capture (fswebcam)**.
- Implement **video recording (GStreamer)**.
- Develop a **logging system**.

🔹 **Phase 3:**

- Implement **functional tests (feature validation)**.
- Implement **negative tests (error handling)**.
- Implement **edge cases tests**.

🔹 **Phase 4:**

- Support for **multiple camera selection**.
- Wrap solution in **Docker**.

---

# 6. Conclusion

This design ensures a **structured, automated, and extensible testing framework** for laptop cameras, setting the foundation for testing additional hardware components in the future.