**1. Introduction *(Camera On)***

“Hi everyone!  
My name is Purna and I want to quickly walk you through my project, **‘Static Code Analyzer’** — a lightweight Python tool designed to analyze .py source files for common issues and code smells.

This project helps developers detect **performance issues** in Python code — such as inefficient loops, memory usage, and blocking operations.

It provides **real-time feedback** through a simple graphical interface, helping developers write cleaner and more efficient code by flagging bad practices early.

It also lays the foundation for **AI-assisted code review**, making the tool smarter over time.”

**✅ 2. Running the Application (Local Execution) *(Screen Share – Terminal)***

“To run the app locally:

1. Clone the repository:

git clone https://github.com/pbthapa123/static\_code\_analyzer.git

cd static\_code\_analyzer

1. Run the GUI:

python gui.py

This project requires **Python 3.8 or above**.  
No third-party libraries are required — it uses the built-in tkinter module for the graphical interface.”

**✅ 3. Features Overview *(Screen Share – GUI)***

“Once the GUI opens, you can upload any .py file using the file browser.

Here’s a quick tour of the layout:

* A **File Upload Section**
* An **Analyze Button**
* And an **Output Console** that displays flagged issues

The tool uses static rules to catch common code smells, such as:

* ❌ time.sleep() – Blocking operations that can freeze execution
* ❌ Missing function docstrings – Makes code harder to understand
* ❌ Unreachable code – Like anything after a return, break, or continue
* ❌ Dangerous variable reassignments – Changing a variable’s type mid-function
* ❌ Improper use of is vs == – Common logic bug in Python

The results are shown in both the **console and the GUI**, with **line numbers and clear messages** to guide developers in fixing the issues.

This entire GUI was created using tkinter, keeping it lightweight and responsive.”

**✅ 4. Code Example (Demo) *(Screen Share – Sample File + Live Output)***

“Here’s an example Python file I used to test the analyzer:

def read\_file\_slow():

sleep(2)

return

print("Unreachable")

def build\_list\_badly():

numbers = [i for i in range(10000)]

numbers = numbers + 1

When I analyze this code, the tool correctly flags:

* The sleep(2) as a **blocking operation**
* The print() line as **unreachable code**
* The numbers variable being reassigned to a different type
* And both functions **missing docstrings**

This demonstrates how the tool provides immediate, useful feedback.”

**✅ 5. GUI Experience *(Screen Share – GUI Interaction)***

“The GUI is designed to be **clean and beginner-friendly**.

You just:

* Upload a .py file
* Click the ‘Analyze’ button
* And results are displayed instantly

It’s a **lightweight tool**, but very **extendable** for future enhancements — especially for developers and students who want a fast code-checking experience.”

**✅ 6. Executable for Testing (Ease of Execution)**

“I also explored creating a standalone .exe file using PyInstaller, but ran into compatibility issues with **Python 3.13**, which is still in preview.

So for now, the best way to run the app is:

python gui.py

No installation or build process is needed.

✅ *Pro Tip:*  
If you’d like to recreate the .exe for distribution, using **Python 3.10** is ideal — it’s a stable version and works well with PyInstaller.”

**✅ 7. Planned Future Work (ML + Feature Expansion) *(Camera On + Visual Slide Optional)***

“While the current version focuses on static, rule-based analysis, I plan to **integrate machine learning** to detect more complex patterns in future updates.

Some planned AI features include:

* Detecting **memory-heavy patterns**, such as large list or dictionary creations
* Identifying **inefficient loop constructs**, like nested loops or long iterations
* Learning from clean codebases to offer **intelligent, context-aware suggestions**

This would take the tool from a rule-based checker to a **smart assistant** for developers.”

**✅ 8. Conclusion *(Camera On)***

“So that’s my project: **Static Code Analyzer**!

✅ It performs essential static checks  
✅ Gives clear, actionable feedback  
✅ Has a simple and clean GUI  
✅ And sets the stage for future **AI-driven enhancements**

Thanks so much for watching!