Steps for Dynamic Analysis Using GDB:

1. Copy the Code and Compile

2. Debug Using GDB

Once inside GDB, I set breakpoints at key points like the login and registry functions,

(gdb) break login

(gdb) break registry

Then, I started the program with:

(gdb) run

3. Analyzing Code Execution

Upon reaching the login and registry sections, I stepped through the code using the `next` and `step` commands to observe how variables like `UserName`, `Password`, `newName`, and `newPass` were being handled.

- I noticed that when entering the username and password in the login function, the program sometimes failed to open the `users.txt` file correctly, likely due to a case-sensitivity issue (the file is opened as `users.txt` in the login function and as `Users.txt` in the registry function).

- The input validation loop (`while`) in both the login and registry sections functioned as expected, though no clear feedback was provided to the user when a file-related error occurred.

4. Dynamic Analysis Output

The program output confirmed that when registering a new user, the file was not being opened properly in the login section due to the file name mismatch.

Resolution:

I modified both the login and registry functions to ensure consistent file handling by using the correct file name (`users.txt` in both cases).

After resolving the issue, the program correctly handled login attempts and user registrations.

How PVS-Studio and GDB Complement Each Other:

1. PVS-Studio(Static Analysis Tool):

PVS-Studio analyzes the code without executing it. It detects potential issues like memory leaks, buffer overflows, and undefined behavior by examining the code's structure and logic.

PVS-Studio would help identify issues such as uninitialized variables or risky string handling early in the development process, before running the code.

2. GDB (Dynamic Analysis Tool):

GDB allows step-by-step execution of the code, making it possible to observe how the program behaves at runtime. This is particularly useful for debugging crashes, segmentation faults, or logical errors.

GDB helps in tracking down the exact point of failure during execution, such as incorrect file handling or incorrect control flow due to runtime conditions.

Together, PVS-Studio finds static issues in the code without execution, while GDB helps analyze and fix dynamic issues during program execution.