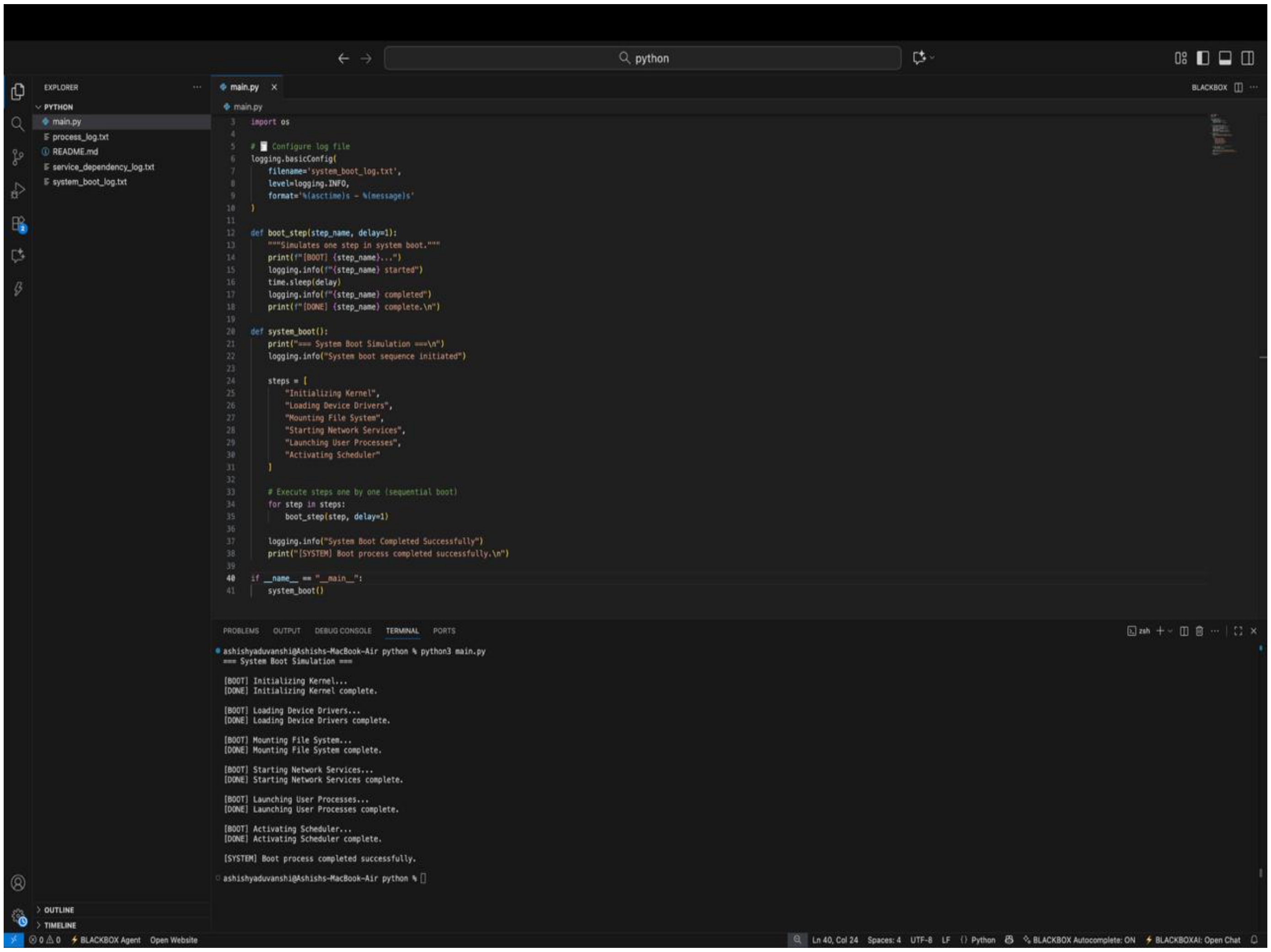
Sub-Task 1: Initialize the logging configuration

Objective: Set up the logging system to log messages with timestamps and process names.

```
import logging

# Setup logger

logging.basicConfig(

    filename='process_log.txt',

    level=logging.INFO,

    format='%(asctime)s - %(processName)s - %(message)s'

)
```



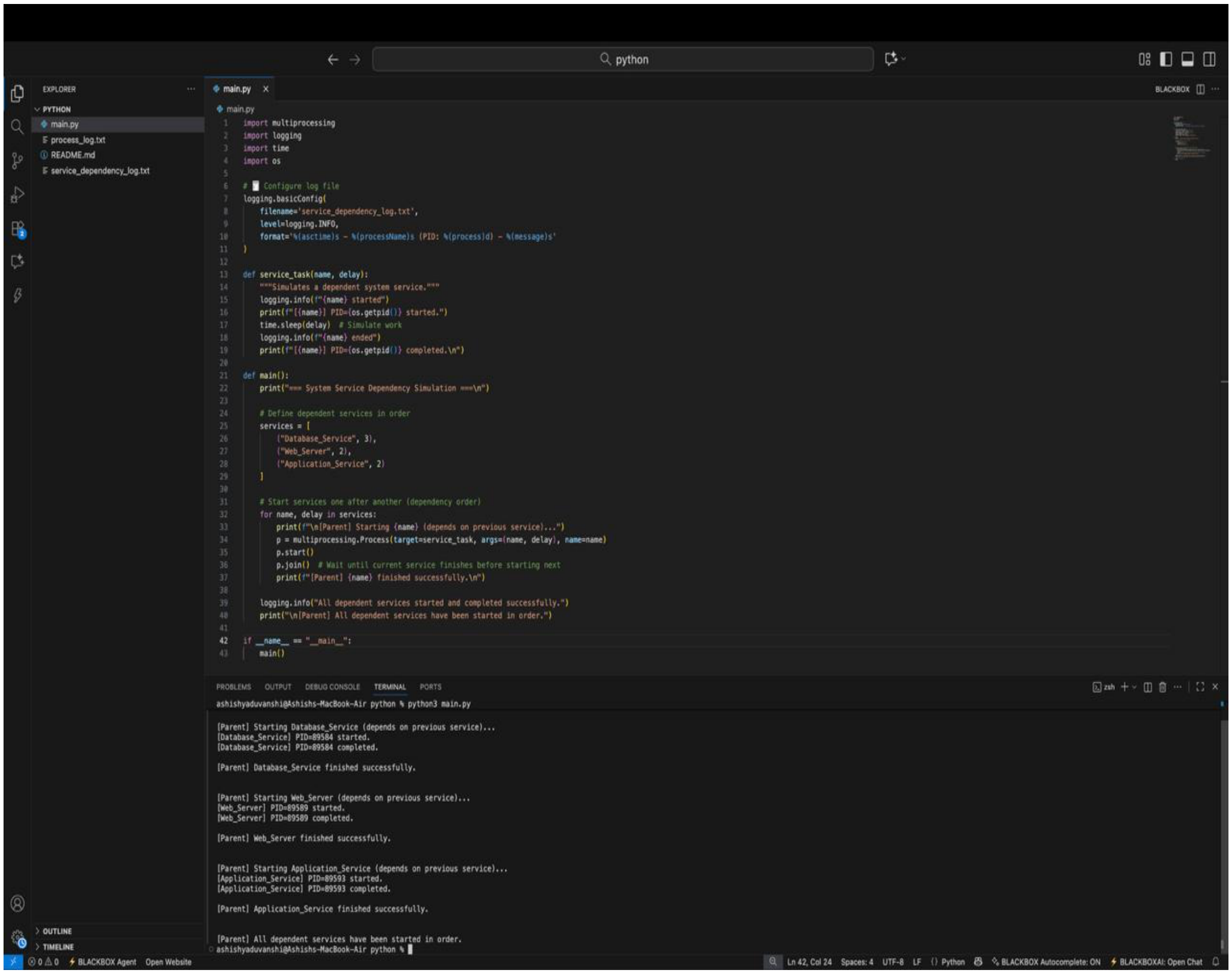
```python
import os

# 📄 Configure log file
logging.basicConfig(
    filename='system_boot_log.txt',
    level=logging.INFO,
    format='%(asctime)s - %(message)s'
)

def boot_step(step_name, delay=1):
    """Simulates one step in system boot."""
    print(f"[BOOT] {step_name}...")
    logging.info(f"{step_name} started")
    time.sleep(delay)
    logging.info(f"{step_name} completed")
    print(f"[DONE] {step_name} complete.\n")

def system_boot():
    print("=== System Boot Simulation ===\n")
    logging.info("System boot sequence initiated")

    steps = [
        "Initializing Kernel",
        "Loading Device Drivers",
        "Mounting File System",
        "Starting Network Services",
        "Launching User Processes",
        "Activating Scheduler"
    ]

    # Execute steps one by one (sequential boot)
    for step in steps:
        boot_step(step, delay=1)

    logging.info("System Boot Completed Successfully")
    print("[SYSTEM] Boot process completed successfully.\n")

if __name__ == "__main__":
    system_boot()
```

```
ashishyaduvanshi@Ashishs-MacBook-Air python % python3 main.py
=== System Boot Simulation ===

[BOOT] Initializing Kernel...
[DONE] Initializing Kernel complete.

[BOOT] Loading Device Drivers...
[DONE] Loading Device Drivers complete.

[BOOT] Mounting File System...
[DONE] Mounting File System complete.

[BOOT] Starting Network Services...
[DONE] Starting Network Services complete.

[BOOT] Launching User Processes...
[DONE] Launching User Processes complete.

[BOOT] Activating Scheduler...
[DONE] Activating Scheduler complete.

[SYSTEM] Boot process completed successfully.

ashishyaduvanshi@Ashishs-MacBook-Air python %
```

Sub-Task 2: Define a function that simulates a process task

Objective: Write a function that mimics the work of a system process.

```
import time

# Dummy function to simulate a task

def system_process(task_name):

    logging.info(f"{task_name} started")

    time.sleep(2)  # Simulate task delay

    logging.info(f"{task_name} ended")
```



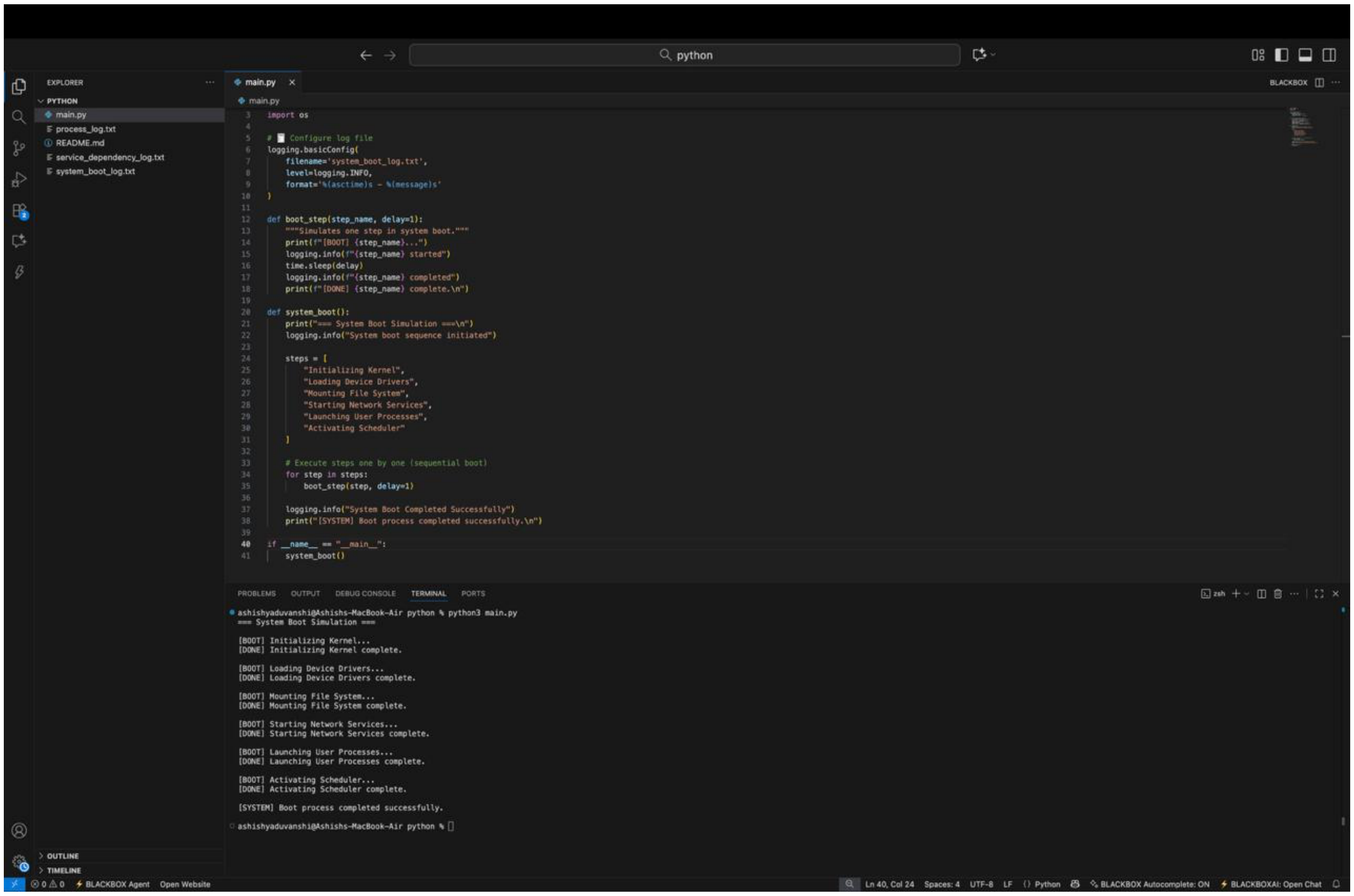
```
import multiprocessing
import logging
import time
import os

# Configure log file
logging.basicConfig(
    filename='service_dependency_log.txt',
    level=logging.INFO,
    format='%(asctime)s - %(processName)s (PID: %(process)d) - %(message)s'
)

def service_task(name, delay):
    """Simulates a dependent system service."""
    logging.info(f"{name} started")
    print(f"[{name}] PID={os.getpid()} started.")
    time.sleep(delay)  # Simulate work
    logging.info(f"{name} ended")
    print(f"[{name}] PID={os.getpid()} completed.\n")

def main():
    print("=== System Service Dependency Simulation ===\n")

    # Define dependent services in order
    services = [
        ("Database_Service", 3),
        ("Web_Server", 2),
        ("Application_Service", 2)
    ]

    # Start services one after another (dependency order)
    for name, delay in services:
        print(f"\n[Parent] Starting {name} (depends on previous service)...")
        p = multiprocessing.Process(target=service_task, args=(name, delay), name=name)
        p.start()
        p.join()  # Wait until current service finishes before starting next
        print(f"[Parent] {name} finished successfully.\n")

    logging.info("All dependent services started and completed successfully.")
    print("\n[Parent] All dependent services have been started in order.")

if __name__ == "__main__":
    main()
```

```
ashishyaduvanshi@Ashishs-MacBook-Air python % python3 main.py

[Parent] Starting Database_Service (depends on previous service)...
[Database_Service] PID=89584 started.
[Database_Service] PID=89584 completed.

[Parent] Database_Service finished successfully.


[Parent] Starting Web_Server (depends on previous service)...
[Web_Server] PID=89589 started.
[Web_Server] PID=89589 completed.

[Parent] Web_Server finished successfully.


[Parent] Starting Application_Service (depends on previous service)...
[Application_Service] PID=89593 started.
[Application_Service] PID=89593 completed.

[Parent] Application_Service finished successfully.


[Parent] All dependent services have been started in order.
ashishyaduvanshi@Ashishs-MacBook-Air python % 
```

Sub-Task 3: Create at least two processes and start them concurrently

Objective: Use the multiprocessing module to initiate parallel tasks.

```
import multiprocessing

if __name__ == '__main__':

    print("System Starting...")

    # Create processes

    p1 = multiprocessing.Process(target=system_process, args=('Process-1',))

    p2 = multiprocessing.Process(target=system_process, args=('Process-2',))

    # Start processes

    p1.start()

    p2.start()
```



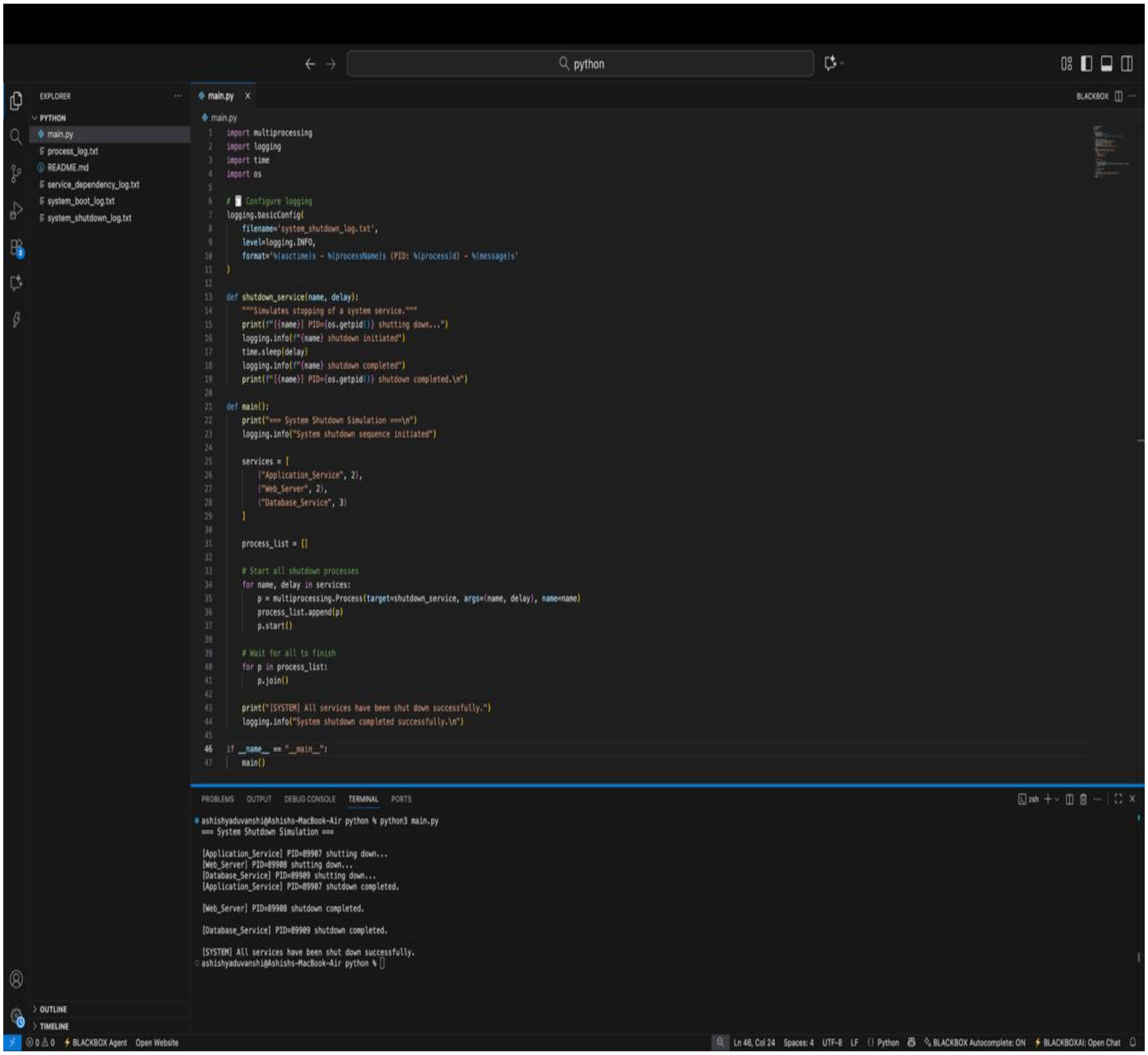
```
import os

# 📄 Configure log file
logging.basicConfig(
    filename='system_boot_log.txt',
    level=logging.INFO,
    format='%(asctime)s - %(message)s'
)

def boot_step(step_name, delay=1):
    """Simulates one step in system boot."""
    print(f"[BOOT] {step_name}...")
    logging.info(f"{step_name} started")
    time.sleep(delay)
    logging.info(f"{step_name} completed")
    print(f"[DONE] {step_name} complete.\n")

def system_boot():
    print("=== System Boot Simulation ===\n")
    logging.info("System boot sequence initiated")

    steps = [
        "Initializing Kernel",
        "Loading Device Drivers",
        "Mounting File System",
        "Starting Network Services",
        "Launching User Processes",
        "Activating Scheduler"
    ]

    # Execute steps one by one (sequential boot)
    for step in steps:
        boot_step(step, delay=1)

    logging.info("System Boot Completed Successfully")
    print("[SYSTEM] Boot process completed successfully.\n")

if __name__ == "__main__":
    system_boot()
```

```
ashishyaduvanshi@Ashishs-MacBook-Air python % python3 main.py
=== System Boot Simulation ===

[BOOT] Initializing Kernel...
[DONE] Initializing Kernel complete.

[BOOT] Loading Device Drivers...
[DONE] Loading Device Drivers complete.

[BOOT] Mounting File System...
[DONE] Mounting File System complete.

[BOOT] Starting Network Services...
[DONE] Starting Network Services complete.

[BOOT] Launching User Processes...
[DONE] Launching User Processes complete.

[BOOT] Activating Scheduler...
[DONE] Activating Scheduler complete.

[SYSTEM] Boot process completed successfully.

ashishyaduvanshi@Ashishs-MacBook-Air python %
```

Sub-Task 4: Ensure proper termination and verify logs

Objective: Wait for processes to complete and confirm the shutdown.

```
# Wait for processes to complete
p1.join()
p2.join()
print("System Shutdown.")
```



```python
import multiprocessing
import logging
import time
import os

# ✅ Configure logging
logging.basicConfig(
    filename='system_shutdown_log.txt',
    level=logging.INFO,
    format='%(asctime)s - %(processName)s (PID: %(process)d) - %(message)s'
)

def shutdown_service(name, delay):
    """Simulates stopping of a system service."""
    print(f"[{name}] PID={os.getpid()} shutting down...")
    logging.info(f"{name} shutdown initiated")
    time.sleep(delay)
    logging.info(f"{name} shutdown completed")
    print(f"[{name}] PID={os.getpid()} shutdown completed.\n")

def main():
    print("=== System Shutdown Simulation ===\n")
    logging.info("System shutdown sequence initiated")

    services = [
        ("Application_Service", 2),
        ("Web_Server", 2),
        ("Database_Service", 3)
    ]

    process_list = []

    # Start all shutdown processes
    for name, delay in services:
        p = multiprocessing.Process(target=shutdown_service, args=(name, delay), name=name)
        process_list.append(p)
        p.start()

    # Wait for all to finish
    for p in process_list:
        p.join()

    print("[SYSTEM] All services have been shut down successfully.")
    logging.info("System shutdown completed successfully.\n")

if __name__ == "__main__":
    main()
```

```
ashishyaduvanshi@Ashishs-MacBook-Air python % python3 main.py
=== System Shutdown Simulation ===

[Application_Service] PID=89907 shutting down...
[Web_Server] PID=89908 shutting down...
[Database_Service] PID=89909 shutting down...
[Application_Service] PID=89907 shutdown completed.

[Web_Server] PID=89908 shutdown completed.

[Database_Service] PID=89909 shutdown completed.

[SYSTEM] All services have been shut down successfully.
ashishyaduvanshi@Ashishs-MacBook-Air python % 
```