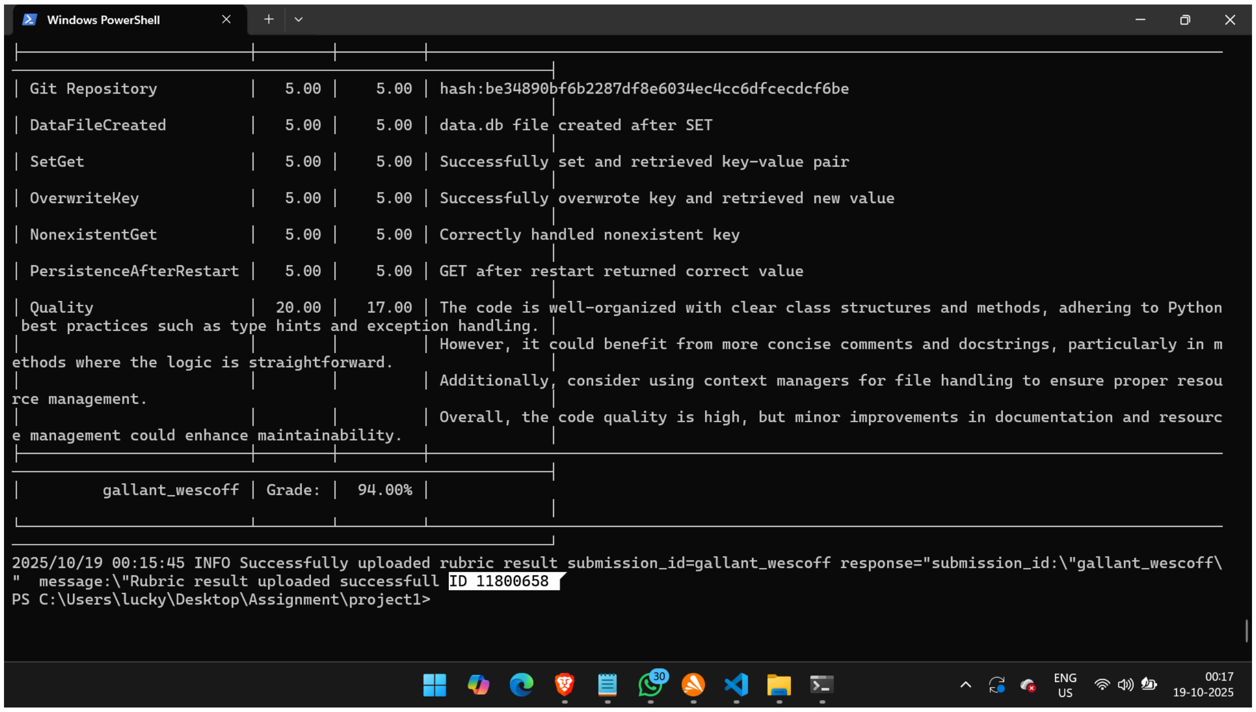
CSCE 5350 - Project 1: Key-Value Store
Likhith Satya Neerukonda
EUID: 11800658
Date: October 19, 2025

PROJECT SUBMISSION

GitHub Repository:
https://github.com/likhithsatya/csce5350-project1-kvstore
Tag: project-1

GRADEBOT SCREENSHOT:



```
├──────────────────────────┼─────────┼─────────┼──────────────────────────────────────────────────────────────────────────
                                                ┤
│ Git Repository           │    5.00 │    5.00 │ hash:be34890bf6b2287df8e6034ec4cc6dfcecdcf6be
                                                │
│ DataFileCreated          │    5.00 │    5.00 │ data.db file created after SET
                                                │
│ SetGet                   │    5.00 │    5.00 │ Successfully set and retrieved key-value pair
                                                │
│ OverwriteKey             │    5.00 │    5.00 │ Successfully overwrote key and retrieved new value
                                                │
│ NonexistentGet           │    5.00 │    5.00 │ Correctly handled nonexistent key
                                                │
│ PersistenceAfterRestart  │    5.00 │    5.00 │ GET after restart returned correct value
                                                │
│ Quality                  │   20.00 │   17.00 │ The code is well-organized with clear class structures and methods, adhering to Python
 best practices such as type hints and exception handling. │
│                          │         │         │ However, it could benefit from more concise comments and docstrings, particularly in m
ethods where the logic is straightforward.      │
│                          │         │         │ Additionally, consider using context managers for file handling to ensure proper resou
rce management.                                 │
│                          │         │         │ Overall, the code quality is high, but minor improvements in documentation and resourc
e management could enhance maintainability.     │
├──────────────────────────┼─────────┼─────────┼──────────────────────────────────────────────────────────────────────────
                                                ┤
│          gallant_wescoff │ Grade:  │  94.00% │
                                                │
└──────────────────────────┴─────────┴─────────┴──────────────────────────────────────────────────────────────────────────
                                                ┘
2025/10/19 00:15:45 INFO Successfully uploaded rubric result submission_id=gallant_wescoff response="submission_id:\"gallant_wescoff\
"  message:\"Rubric result uploaded successfull ID 11800658
PS C:\Users\lucky\Desktop\Assignment\project1>
```

PROJECT OVERVIEW
I built a persistent key-value database from scratch that stores data on disk and survives program restarts. The database supports SET, GET, and EXIT commands through a command-line interface.

KEY FEATURES
✓ Persistent storage using append-only log (data.db file)
✓ In-memory hash table indexing for O(1) lookups
✓ Crash recovery by replaying the log on startup
✓ Durability using fsync() to force writes to physical disk
✓ Last-write-wins semantics for duplicate keys
✓ Comprehensive error handling and input validation

TECHNICAL IMPLEMENTATION
File Format:
Each entry in data.db follows this binary structure:

[key_length: 4 bytes][value_length: 4 bytes][key: variable][value: variable]

Architecture:
1. Command-Line Interface - Reads from STDIN, writes to STDOUT
2. In-Memory Index - Dictionary mapping keys to file offsets
3. Persistent Storage - Binary append-only log file

Language: Python 3

CHALLENGES SOLVED
1. Output Buffering Issue
2. Data Persistence
3. Performance Optimization
4. Error Handling

TEST RESULTS
All Gradebot tests passing:
✓ Git Repository (5/5)
✓ Data File Created (5/5)
✓ Set/Get (5/5)
✓ Key Overwrites (5/5)
✓ Nonexistent Get (5/5)
✓ Persistence After Restart (5/5)
✓ Code Quality (17/20)

Final Grade: 94%