Project Report: Shell Script Implementation

1.	Project Name	Simple Local Voting System (CLI)
2	Name	Khushi Mehrotra
3	SAP Id	590028278
4	Name	Aditya Tomar
5	SAP Id	590028723

Project Name	Simple Local Voting System (CLI)
Technology	GNU Bash Shell Script (Linux/Unix)
Date	December 2, 2025
Goal	To create a basic, text-based, local system for vote tallying using foundational shell scripting commands.

1. Problem Statement

The goal of this project was to develop a rudimentary, non-networked application capable of allowing a user to cast a vote for one of several predefined candidates, store the votes and display the calculated results. The primary constraint was that the entire system must be implemented using a single **Linux shell script** (Bash), leveraging common system utilities for data storage, processing, and display.

The system needed to address the following functional requirements:

1. Provide a text-based menu interface for user interaction.
2. Allow a user to select a candidate and record the vote to a persistent file.
3. Implement a basic concurrency control mechanism (file locking) to prevent data corruption during simultaneous write operations.
4. Calculate and display the current vote tally for all candidates in a clear, sorted format.

2. Algorithm

The core functionality is split into three main components: a menu loop, the vote casting function, and the results display function.

A. Vote Casting (cast_vote function)

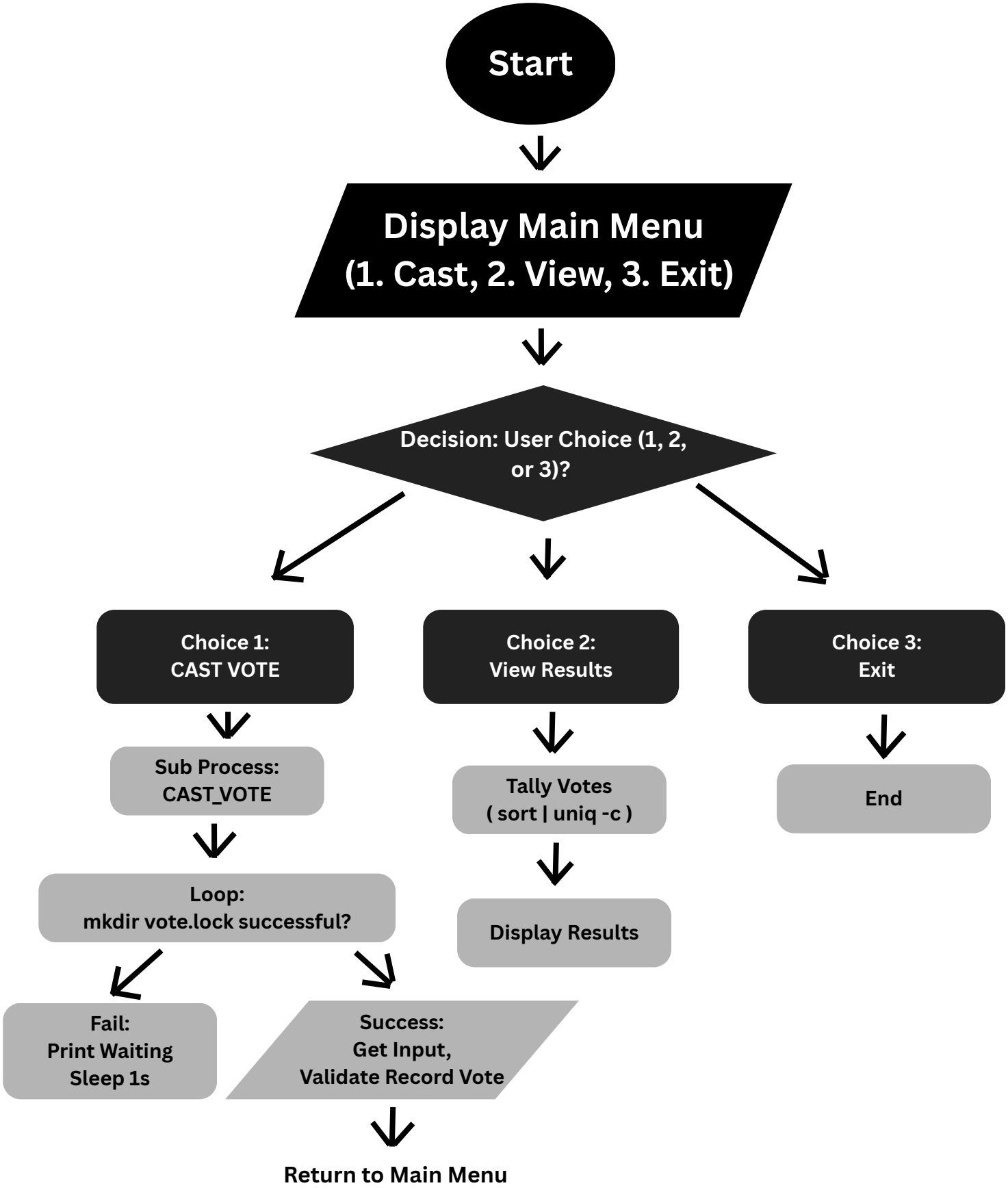
1. **Acquire Lock:** Attempt to create a unique directory (vote.lock).
   - **IF Successful:** Proceed.
   - **IF Failed:** The lock is held by another process. Print a waiting message and sleep 1 second before retrying (Go to Step 1).
2. **Display Options:** Print the list of available candidates with corresponding numeric indices (1, 2, 3, 4).
3. **Get Input:** Read the user's selected candidate number.
4. **Validate:** Check if the input is a valid integer between 1 and 4.
5. **Record Vote:** If valid, append the full candidate name to the votes.txt file (echo "$candidate" >> "$VOTE_FILE").
6. **Release Lock:** Remove the lock directory (rmdir "$LOCK_FILE").
7. **Error Handling:** If invalid, notify the user and do not record a vote.

B. Results Tallying (show_results function)

1. **Check Data:** Verify that the votes.txt file exists and contains data.
2. **Process Data:** Pipe the contents of votes.txt through a sequence of commands:
   - cat "$VOTE_FILE": Reads all recorded votes.
   - | sort: Sorts the list alphabetically, grouping identical candidate names together.
   - | uniq -c: Counts the number of times each unique candidate name appears.
   - | sort -nr: Re-sorts the results numerically in reverse order (highest count first).
3. **Display:** Loop through the sorted output, printing the candidate name and their corresponding vote count.
4. **Total Count:** Use wc -l < "$VOTE_FILE" to determine and display the total number of votes cast.

3. Flowchart

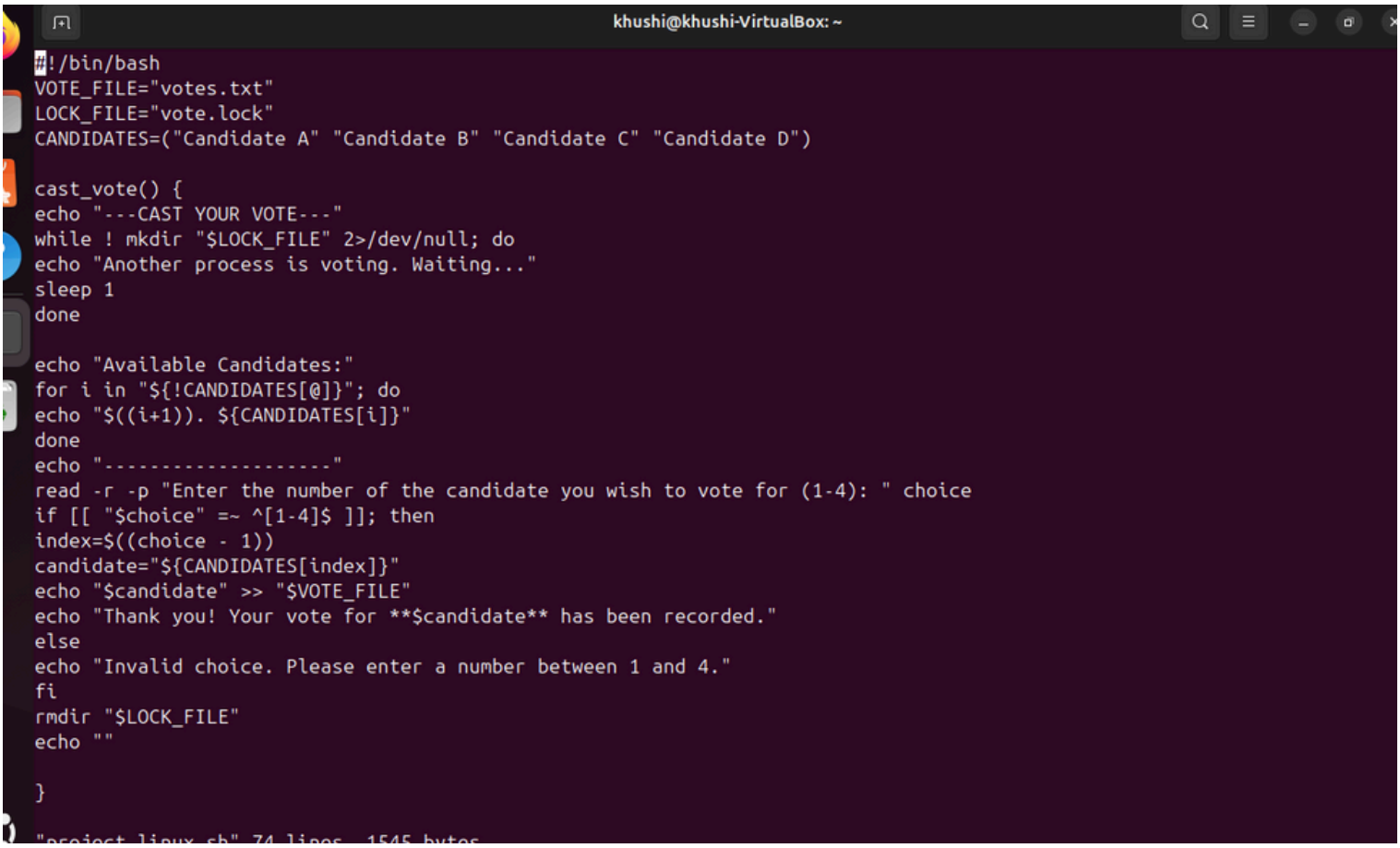
This flowchart illustrates the main execution path and the locking logic in the vote casting function.

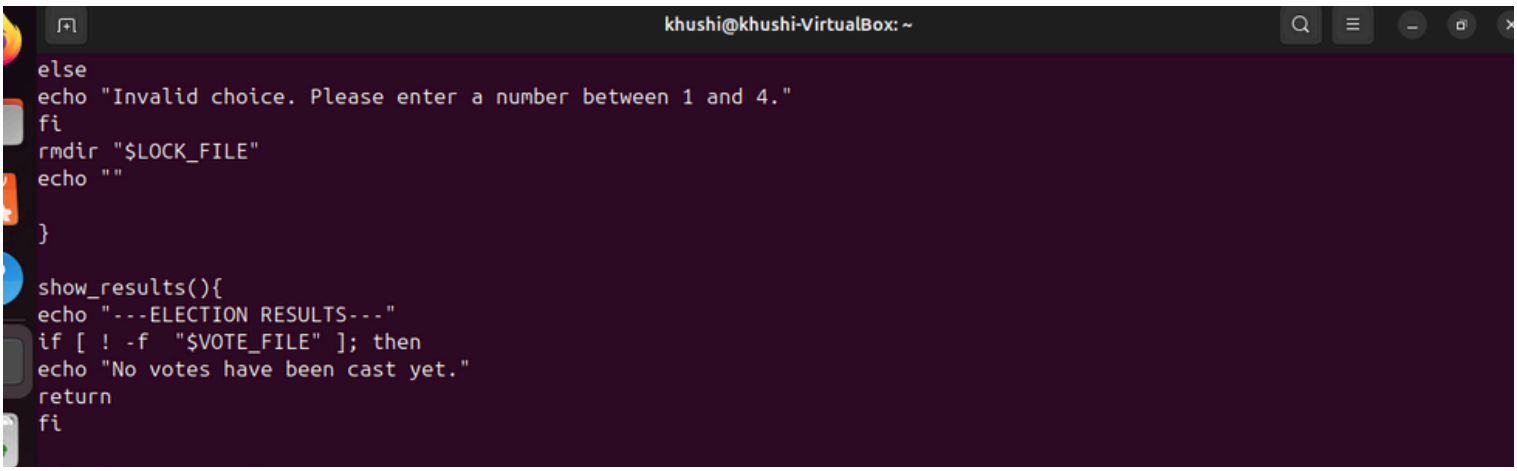


4. Problems Encountered and Solutions

Problem	Description	Solution Implemented
Concurrency/Data Corruption	Multiple users running the script simultaneously could attempt to write to votes.txt at the same moment, leading to garbled or lost data.	**File Locking:** Implemented a non-race condition lock using mkdir. Since mkdir is an atomic operation, only one process can successfully create the vote.lock directory. The other processes wait (using sleep 1) until the directory is removed by the first process via rmdir.
Inaccurate Vote Tally	Simply reading the file and counting lines would not separate the votes by candidate.	**Piping Utilities:** Utilized the powerful Linux pipeline (`
User Experience (Waiting)	If a user is waiting for the lock, the script could continuously cycle, consuming CPU resources unnecessarily.	**The sleep Command:** Introduced sleep 1 inside the locking loop. This pauses the script for one second before retrying the lock, significantly reducing CPU load during concurrent access.
Input Validation	Entering text or numbers outside the range (e.g., "5" or "exit") would cause incorrect behavior or errors.	**Regex Validation:** Used the Bash built-in conditional [["$choice" =~ ^[1-4]$]] to ensure the input was a single digit between 1 and 4 before attempting to process the vote.

5. Snippets of Code

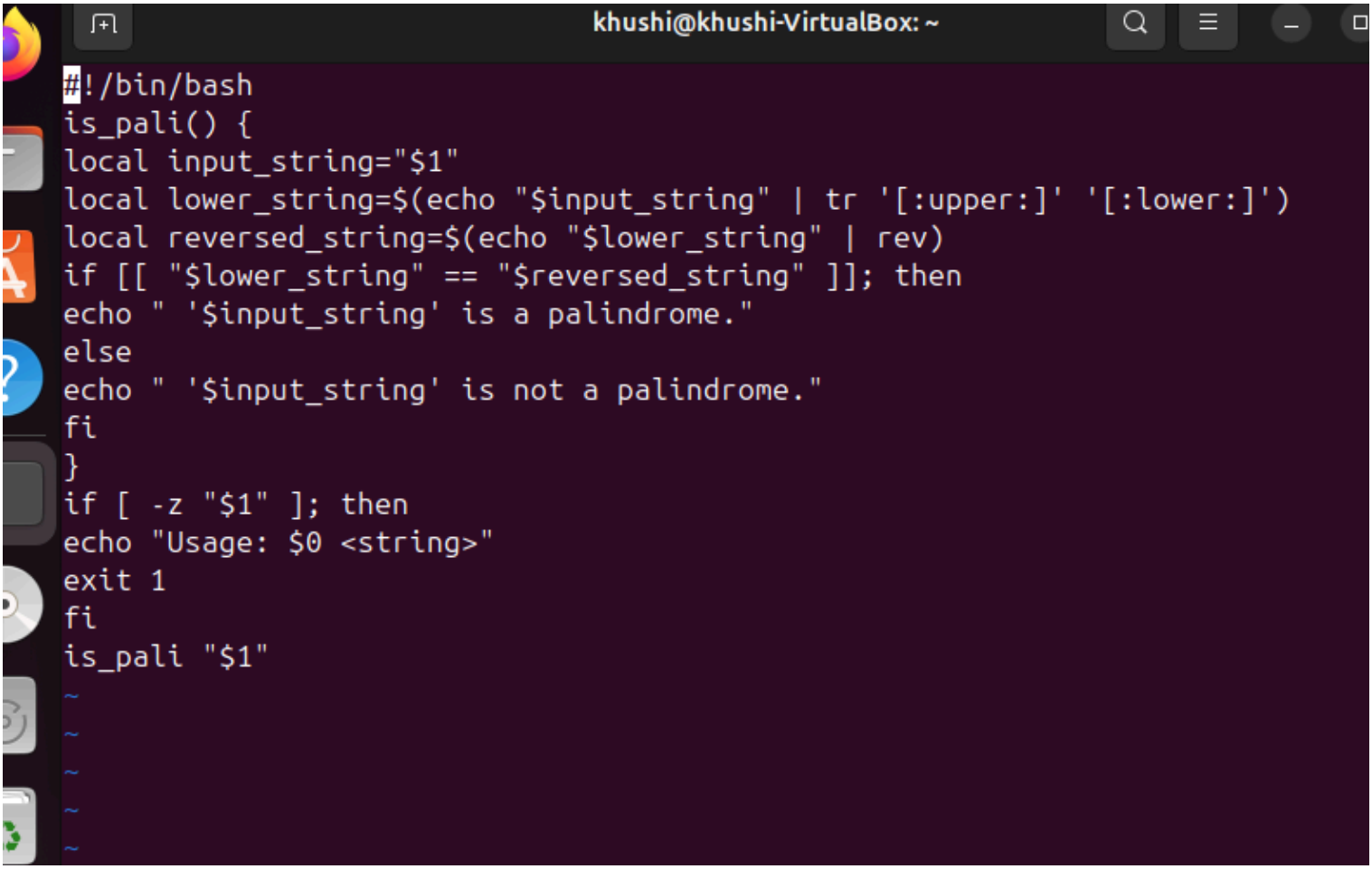




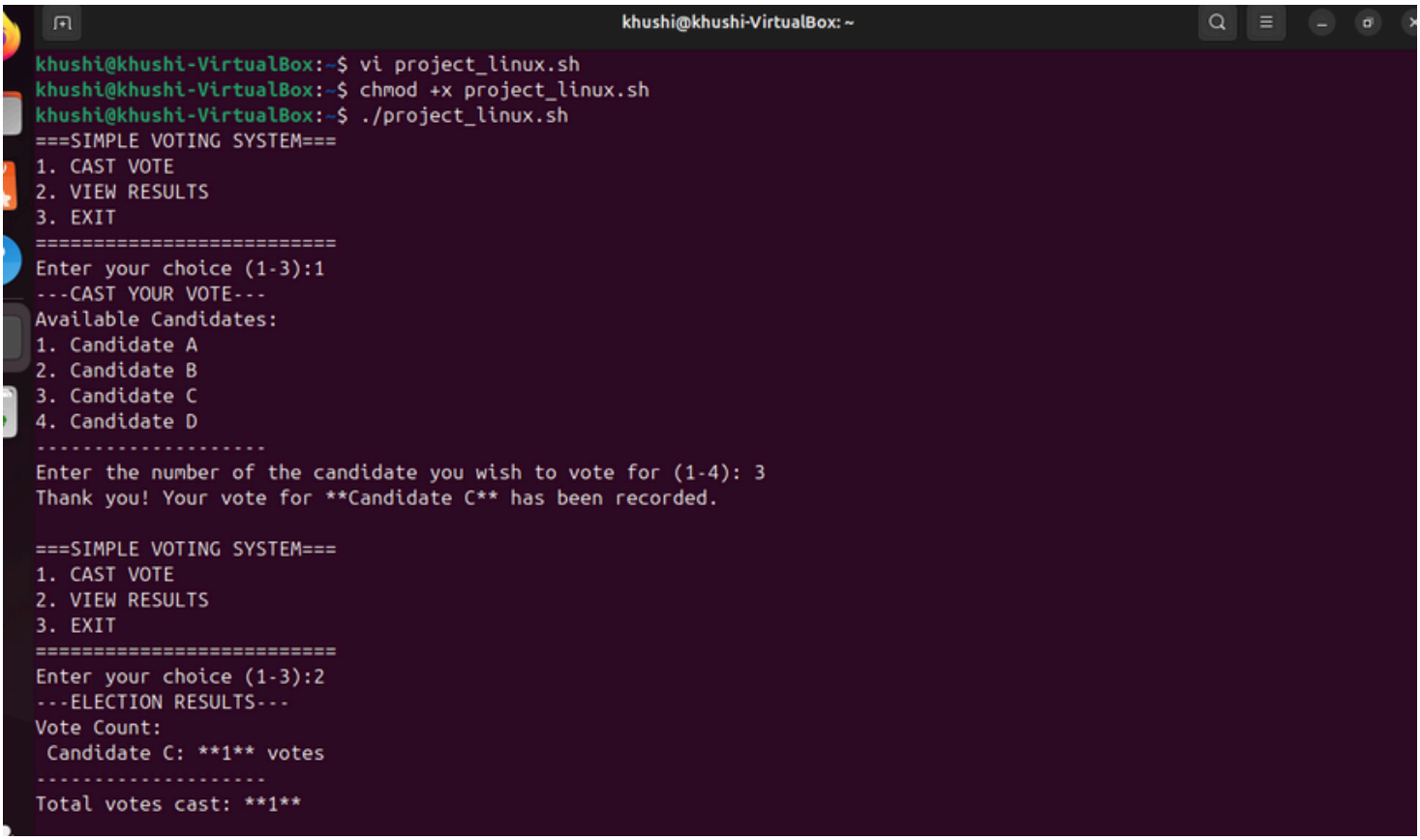
```
echo "Vote Count:"
cat "$VOTE_FILE" | sort | uniq -c | sort -nr | while read -r count name; do
echo " $name: **$count** votes"
done

echo "--------------------"
total_votes=$(wc -l < "$VOTE_FILE")
echo "Total votes cast: **$total_votes**"
echo ""
}

while true; do
echo "===SIMPLE VOTING SYSTEM==="
echo "1. CAST VOTE"
echo "2. VIEW RESULTS"
echo "3. EXIT"
```

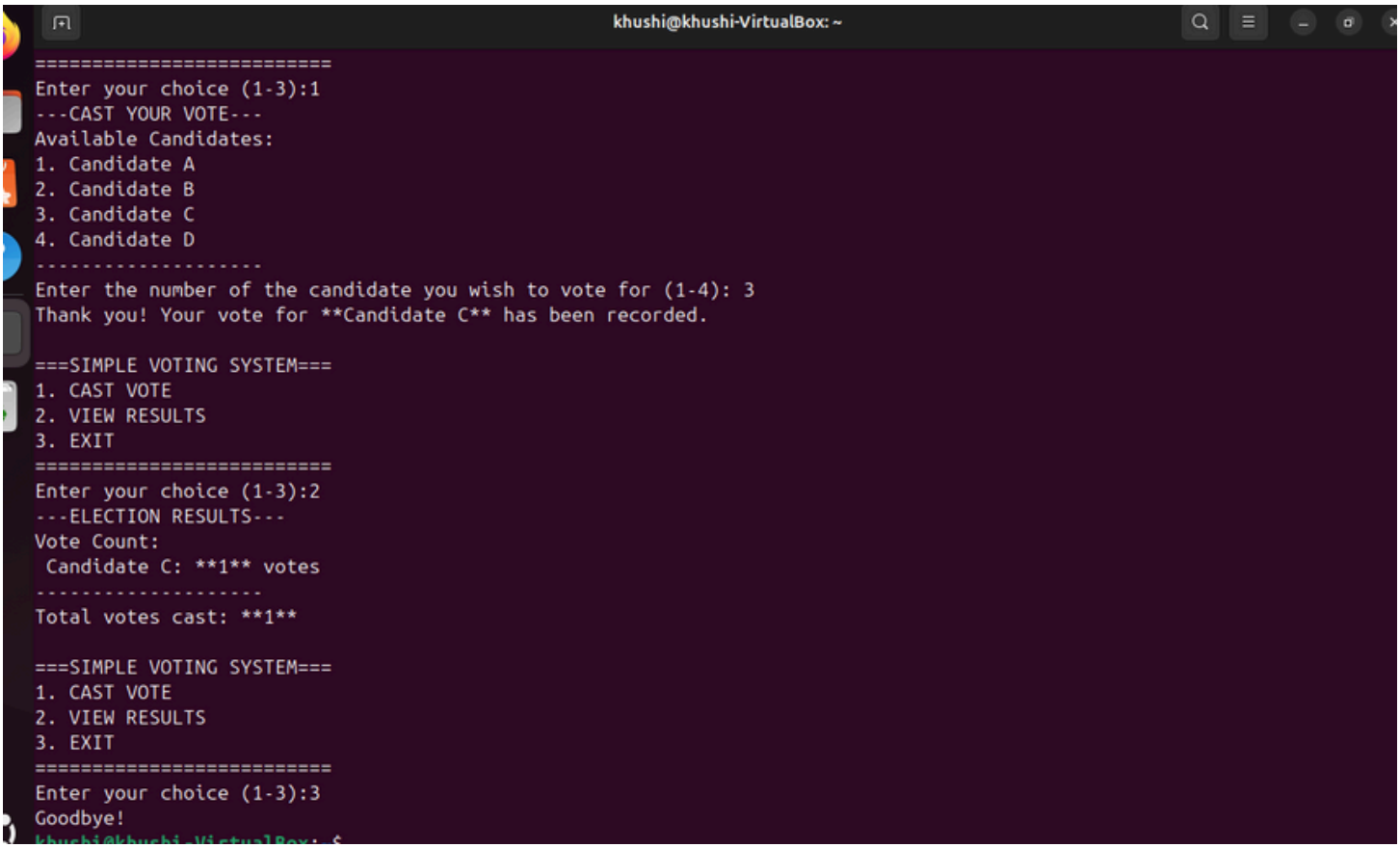


6. Output


khushi@khushi-VirtualBox: ~
khushi@khushi-VirtualBox:~$ vi project_linux.sh
khushi@khushi-VirtualBox:~$ chmod +x project_linux.sh
khushi@khushi-VirtualBox:~$ ./project_linux.sh
===SIMPLE VOTING SYSTEM===
1. CAST VOTE
2. VIEW RESULTS
3. EXIT
==========
Enter your choice (1-3):1
---CAST YOUR VOTE---
Available Candidates:
1. Candidate A
2. Candidate B
3. Candidate C
4. Candidate D

Enter the number of the candidate you wish to vote for (1-4): 3
Thank you! Your vote for **Candidate C** has been recorded.
===SIMPLE VOTING SYSTEM===
1. CAST VOTE
2. VIEW RESULTS
3. EXIT
==========
Enter your choice (1-3):2
---ELECTION RESULTS---
Vote Count:
Candidate C: **1** votes

Total votes cast: **1**


khushi@khushi-VirtualBox: ~
==========
Enter your choice (1-3):1
---CAST YOUR VOTE---
Available Candidates:
1. Candidate A
2. Candidate B
3. Candidate C
4. Candidate D

Enter the number of the candidate you wish to vote for (1-4): 3
Thank you! Your vote for **Candidate C** has been recorded.
===SIMPLE VOTING SYSTEM===
1. CAST VOTE
2. VIEW RESULTS
3. EXIT
==========
Enter your choice (1-3):2
---ELECTION RESULTS---
Vote Count:
Candidate C: **1** votes

Total votes cast: **1**
===SIMPLE VOTING SYSTEM===
1. CAST VOTE
2. VIEW RESULTS
3. EXIT
==========
Enter your choice (1-3):3
Goodbye!