□ Low-Level Design (LLD) – Simple Voting System

Difficulty Level: Basic | Total Marks: 20

Standards Followed: 4 Functions | 4 Visible Test Cases | 2 Hidden Test Cases

——

□ Summary of Corrections (Based on SME Feedback)

□ Used NumPy arrays to store and validate vote IDs.

□ Added vote count using np.unique with return_counts=True.

□ Winner determined by highest count index.

□ All validation checks ensure numeric-only input.

□ Standardized input/output structure.

□ Concepts Tested

□ NumPy Array Creation

□ Unique Element Counting

□ Frequency-based Winner Detection

□ Input Validation

□ Problem Statement

Count votes for multiple candidates and determine the winner using NumPy arrays.

□ Operations

1. Create Vote Array

Creates a NumPy array from vote IDs.

Function Prototype:

def create_vote_array(vote_ids: list) -> np.ndarray

□ Example Input:

[1, 2, 2, 3]

□ Expected Output:

[1, 2, 2, 3]

□ Implementation Flow:

• Convert list to np.ndarray

• Return array

2. Count Votes Per Candidate

Counts number of votes per candidate.

Function Prototype:

def count_votes_per_candidate(vote_array: np.ndarray) -> np.ndarray

 Example Input:

[1, 2, 2, 3]

 Expected Output:

[1, 2, 1]

 Implementation Flow:

• Use np.unique with return_counts

• Return counts array

3. Determine Winner

Returns candidate with highest votes.

Function Prototype:

def determine_winner(vote_array: np.ndarray) -> int

 Example Input:

[1, 2, 2, 3]

 Expected Output:

2

 Implementation Flow:

• Use np.unique to count

• Use np.argmax to identify winner

4. Validate Vote Array

Checks that input array has only integers 1–5.

Function Prototype:

def validate_vote_array(vote_array: np.ndarray) -> bool

 Example Input:

[1, 2, 2, 3]

 Expected Output:

True

 Implementation Flow:

• Check type and range

• Return boolean

Implementation Code

```
import numpy as np

def create_vote_array(vote_ids: list) -> np.ndarray:
    """
```

```
Converts a list of vote IDs into a NumPy array.
"""
    # Convert the vote_ids list into a NumPy array using np.array()
    # Make sure to set dtype to int
    pass
def count_votes_per_candidate(vote_array: np.ndarray) -> np.ndarray:
    """
    Counts the number of votes received by each candidate.
    """
    # Use np.unique() with return_counts=True
    # Return only the counts of each candidate
    pass
def determine_winner(vote_array: np.ndarray) -> int:
    """
    Determines the winner (candidate with the most votes).
    """
    # Use np.unique() to get candidate IDs and their counts
    # Use np.argmax() on counts to find the index of the highest count
    # Return the corresponding candidate ID
    pass
def validate_vote_array(vote_array: np.ndarray) -> bool:
    """
 Validates if the vote array contains only integers from 1 to 5.
    """
    # Check if all values are integers
```

```
    # Use NumPy condition to check if all values are within range 1 to 5
    # Return True if valid, otherwise False
    pass
```

✅ Test Cases and Marks Allocation

Test Case ID	Description	Associated Function	Marks
TC1	Create vote array	create_vote_array	✅ 2.5
TC2	Count votes	count_votes_per_candidate	✅ 2.5
TC3	Determine winner	determine_winner	✅ 2.5
TC4	Validate vote input	validate_vote_array	✅ 2.5
HTC1	Edge vote count	determine_winner	✅ 2.5
HTC2	Invalid vote range	validate_vote_array	✅ 2.5

✅ Visible Test Cases (4)

TC1: Create Vote Array
✅ Input: [1, 2, 2, 3]
✅ Expected Output: [1, 2, 2, 3]

TC2: Vote Counts
✅ Input: [1, 2, 2, 3]
✅ Expected Output: [1, 2, 1]

TC3: Winner Detection
✅ Input: [1, 2, 2, 3]
✅ Expected Output: 2

TC4: Validation
✅ Input: [1, 2, 3]
✅ Expected Output: True

✅ Hidden Test Cases (2)

HTC1: Winner Edge Tie
✅ Input: [1, 2, 1, 2]
✅ Expected Output: 1 or 2

HTC2: Invalid Vote Detected
✅ Input: [0, 6]
✅ Expected Output: False