AI Assisted Coding

Lab Exam:

H.T.NO:2403A51267

Batch:12

Task-1:

Question:

Context:

A small real estate listings platform utility needs a README for onboarding.

Your Task:

From comments, produce README: Overview, Setup, Usage, Tests, Limitations + a CLI example.

Data & Edge Cases:

Module + functions listed in comments.

AI Assistance Expectation:

Use AI to draft structure then refine.

Constraints & Notes:

Include one CLI block.

Sample Input

# module: real estate listings platform utilities  
# functions: parse, validate, export

Sample Output

README with 5 sections and example

Acceptance Criteria: Contains required sections

Prompt:

Generate a README.md with Overview, Setup, Usage (programmatic + CLI), Tests, and Limitations for a Python module with functions parse, validate, export.

Code:

from real\_estate\_utils import parse, validate, export

data = parse("listings.csv")

valid\_listings = validate(data)

export(valid\_listings, "output.json")

Output:

This is the \*final README\* with all 5 required sections and the CLI example.

Task-2:

Question:

Context:

Legacy real estate listings platform code uses nested loops for aggregation.

Your Task:

Refactor to dict.get/defaultdict with type hints.

Data & Edge Cases:

Example: [('a',1),('b',2),('a',3)] -> {'a':4,'b':2}.

AI Assistance Expectation:

Ask AI for refactor and parity tests.

Constraints & Notes:

Typed function signature preferred.

Sample Input

data=[('a',1),('b',2),('a',3)]

Sample Output

{'a':4,'b':2}

Acceptance Criteria: Behavior unchanged

Prompt:

Refactor nested-loop aggregation to a defaultdict with type hints, summing values by key.

Code:

from typing import List, Tuple, Dict

from collections import defaultdict

def aggregate\_listings(data: List[Tuple[str, int]]) -> Dict[str, int]:

result: Dict[str, int] = defaultdict(int)

for key, value in data:

result[key] += value

return dict(result)

# Parity tests

if \_name\_ == "\_main\_":

# Sample input/output

data = [('a', 1), ('b', 2), ('a', 3)]

assert aggregate\_listings(data) == {'a': 4, 'b': 2}

# Edge cases

assert aggregate\_listings([]) == {}

assert aggregate\_listings([('x', 0)]) == {'x': 0}

assert aggregate\_listings([('a', S-1), ('a', 1)]) == {'a': 0}

assert aggregate\_listings([('a', 1), ('b', 2), ('c', 3)]) == {'a': 1, 'b': 2, 'c': 3}

Output:

{'a':4,'b':2}